Software re-use assessment for quality
M. Ramachandran
School of Computing and Mathematical Sciences,
Liverpool John Moores University, Liverpool, L3 3AF, UK

ABSTRACT

Reuse of software components can improve software quality and productivity significantly. This paper mainly concentrates on our approach to reuse assessment, which is a process of identifying and assessing the characteristics of a reusable component based on a set of well-defined reuse guidelines. This paper also proposes a scheme for measuring reuse potential. These techniques have been successfully demonstrated and guidelines are partially automated by a prototype tool called Reuse AnalyseR and Enhancer (RARE) which takes existing components and checks against a set of criteria, provides reuse advice and analysis, and generates assessment reports, and components that are improved for reuse.

INTRODUCTION

Software reuse has promised to improve software productivity and quality, and maintenance of software systems. Existing approaches have concentrated mainly on development with reuse which is a process of producing software systems with reusable components that have not been developed for reuse. Biggerstaff and Perlis [1] discuss a detailed account of the concepts, models
and existing approaches to software reuse. One of the major problem of systematic reuse is the lack of availability of potentially reusable components. Therefore, the objective of this paper is to address the issue of development for reuse, which is the process of developing components that are potentially reusable. This process is a relatively new area in software reuse research that has emerged to address the issue of developing components for reuse and quality [8,9].

To support this process of developing components for reuse, there have been interesting works on reuse guidelines that have emerged to represent the characteristics of reusable components [3-6]. However, these studies have concentrated on highly conceptual ideas, and they emphasise mostly on general advice including documentation and management issues. Therefore, their guidelines are difficult to understand, and are sometimes too general and often not realisable. Our objective was to identify understandable, measurable, objective and automatable reuse guidelines to achieve total quality in component reuse.

In our work we have selected Ada as a language for study because of its support for reuse and quality. The application domain selected for study is Abstract Data Structures (ADS), for which we have proposed an alternative view of reusability to that of Booch [2], whose work is well known in this area and his components (known as Booch's components) have been marketed on a commercial basis for the first time. These investigations are integrated into a set of guidelines which provides objective detailed advice on how to construct reusable components, and are used as an effective technique for reuse assessment and improvement. Our earlier work has concentrated on general issues on development for reuse [8,9].

This paper mainly concentrates on our approach to reuse assessment, which is a process of identifying and assessing reuse attributes of a component automatically based on checking the components against a set of well-defined reuse guidelines. This paper also proposes a scheme for measuring reuse potential. These techniques have been successfully demonstrated and guidelines are partially automated by a prototype tool called Reuse Analyser and Enhancer (RARE) which takes existing components and checks against a set of criteria,
provides reuse advice and analysis, and also generates assessment reports and components that are improved for reuse. RARE can also identify the significance of a guideline because some of the guidelines may not be met by a component. RARE has been tested against a number of Booch's Ada components and the results are promising.

**REUSE ASSESSMENT**

There is growing interest in producing reusable components [2-6]. Hall [10] discusses some of the reuse initiatives in the UK since 1986. However, there is no way of assessing their quality based on a specific set of criteria, which is the main objective of this paper. Practical reuse guidelines can play a major role in the production of potentially reusable components and provide designers to assess and improve the reusability of software components automatically. Existing reuse guidelines emphasise mostly on general advice and often not realisable [3-6]. Our objective was to identify understandable, measurable, objective and realisable reuse guidelines to achieve total quality in component reuse.

Reuse Assessment is a process of assessing components based on the number of guidelines satisfied against the total number of guidelines that are applicable, and then producing an assessment report. We should also take into account, the significance of the guidelines. This is where we need to automate this process. The outcome of this process is to make sure that the components designed for reuse and satisfy some of the key characteristics.

For the purpose of this assessment, we consider component's specification (Ada packages) rather than implementation to support higher level reuse of abstractions. We need to address some of the problems of development for reuse:

- What are the characteristics of a reusable component?
- How do we measure reuse potential?
- How do we represent and assess the characteristics of a reusable component?
The approach described in this article provides a practical solution to some of these problems. We have successfully formulated objective and realisable reuse guidelines to address these problems in a practical framework [8,9]. These guidelines are partially automated in a tool set described in the later section of this paper. Our guidelines fall into a number of classes [8,9]:

- Domain-oriented reusability, which is a process of identifying reusable domain abstractions in a problem domain.

- Design-oriented reuse, which is a process of identifying design principles that support reuse. For example, if the design method is object-oriented then, we need to study its support for reuse such as inheritance, class abstractions and so on.

- Language-oriented reusability, which is a process of utilising language support for reuse effectively.

Our reuse assessment model is based on formulating a number of guidelines on the above categories and assessing components against each of them. In this article we discuss some of the set of criteria that are identified when formulating the practical and objective reuse guidelines. Each of these guidelines represents one or more of the following criteria:

1. Complex structures must be represented as generic components.

2. Language features (in our case Ada selected for study) such as packages, private type for hiding detailed structural implementation, avoid using Ada's use clause and mode 'out' in parameter declarations, and so on. It is possible to identify such characteristics across languages.

3. Design heuristics and rationale must be provided when choosing a particular design strategy. For example, a rationale must be provided for reuse designers when selecting a choice between dynamic structures and static structures. Similarly, in Ada, a rationale must be provided when selecting a choice between private and limited private types.
4. A reusable component must provide operations on,

- object creation
- object termination
- input and output procedures
- exception handling mechanism
- object iterations
- state inquiry
- state change
- object assignment and copy
- error status and messages

There are many more such criteria and practical reuse guidelines formulated by Ramachandran and Sommerville [9]. Now we will consider a scheme for measuring the reuse potential of a component based on a number of guidelines that are satisfied against the total number guidelines that are applicable. Perhaps, not all guidelines carry equal weight, a significant factor (in terms of percentage) is attached to each guideline. This grading system is essential to assess a component for reuse and to report the results to software engineers. Our reuse assessment measures are as given below:

A component may be:

1. Weakly reusable, whose potential for reuse is low which means it satisfies fewer than 50% of the guidelines. It needs more effort to redesign the original component for reuse.

2. Limitedly reusable, whose potential for reuse is high which means it satisfies between 50-70% of the relevant guidelines and needs some effort to improve.

3. Strongly reusable, whose potential for reuse is high which means it satisfies between 70-90% of the relevant guidelines and needs little modification to improve.

4. Immediately reusable, whose potential for reuse is very high which means it satisfies more than 90% of the relevant guidelines and this can be reused as-it-is without any modification.
Manual assessment may be more tedious and may not be accurate. Therefore, tool support is important to assess existing components against these guidelines automatically and to suggest further improvements to modify those components.

**TOOL SUPPORT**

Our approach has been successfully demonstrated and guidelines are partially automated by a prototype tool called Reuse Analyser and Enhancer (RARE) as shown in Figure 1, which takes existing components and checks against a set of criteria, provides reuse advice and analysis, and generates assessment reports and components that are improved for reuse. RARE can also identify the significance of a guideline because some of the guidelines may not be met by a component.

![Figure 1: Reuse Analyzer and Enhancer](image)

One of the major objective of this system is to demonstrate, how well-defined reuse guidelines can be used to automate the process of reuse assessment by providing support for language analysis and domain analysis. For example, this system takes an Ada component specification, assesses it through two analysis phases, estimates its reusability according to how well it
satisfies a set of reuse guidelines and generates a component which is improved for reuse.

The system interacts with the engineer to discover information that can't be determined automatically. The conclusion of this first pass is an estimate of how many guidelines are applicable to the component and how many of these have been breached. The report generator produces a report with all the information that has been extracted about that component and changes that have been made for reuse.

The second pass involves applying domain knowledge to the system. The component templates have been modelled representing static and dynamic structures. Their reusability is assessed by comparing the component with that template. The support provided by the system ensures that the reuse engineer carries out a systematic analysis of the component according to the suggested guidelines. He or she need not be a domain expert. Again, an analysis is produced which allows the engineer to assess how much work is required to improve system reusability.

Reuse assessment is a process of assessing the reuse potential of a component. It depends on the number of reuse guidelines that are satisfied by the component. RARE predicts this and reports to the reuse engineer. RARE measures the reusability strength of a component based on the percentage of guidelines satisfied such as weakly (less than 50%), strongly (50-70%), limitedly (70-90%), immediately reusable (more than 90%) and it takes into account the significance of a guideline (its importance for reuse). For example, let us consider one of our domain guideline which says,

- "For all complex structures, the components should be implemented as a generic package with the element type as a generic parameter".

Say for instance, if a component of complex structures doesn't possess a generic package then the significance of this guideline becomes very important and therefore, the system immediately reports to the reuse engineer that the component is weakly reusable. The system can make such structural modification automatically if the engineer decides so by responding to the dialogue.
In this way, reuse assessment is being done by RARE. The result of the assessment process is obviously arbitrary but it allows implementations to be compared, reuse improvements to be assessed, and it allows the reuse engineer to re-plan well before reusing his components. The system also produces detailed reports for Ada components, including before and after the assessment and improvement. The report generator produces the complete details of a component submitted to the systems in a tabular form that mainly consists of object name, its class, details of all the subprograms including the details of formal parameters and their class, and details of private types, etc.

CONCLUSION

RARE has been tested against a number of Booch's Ada components and the results are promising. Some of his components fail to satisfy a number of guidelines. RARE has demonstrated that it is possible to formulate practical and objective reuse guidelines. can be used for assessing the characteristics of a reusable component and can be used as an effective technique for knowledge representation. This has also proved that it is possible to produce components that are assessed and improved for reuse and quality. This toolset has potential to be a part of a future generation CASE tool. Further investigation is underway on automating complex reuse guidelines and a complete tool support for the development of reusable components. We also believe that the approach described here is equally applicable to other languages, tools, and application domains.

REFERENCES


