Available online at www.sciencedirect.com
SCIENCE DIRECT®
Science of Computer Programming 56 (2005) 99–116
Science of Computer Programming
www.elsevier.com/locate/scico

A technique for automatic component extraction from object-oriented programs by refactoring

Hironori Washizaki[a,*], Yoshiaki Fukazawa[b]

[a] *Research Center for Testbeds and Prototyping, National Instituite of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan*
[b] *Department of Computer Science, Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555, Japan*

Received 29 October 2003; received in revised form 22 September 2004; accepted 22 September 2004
Available online 21 December 2004

Abstract

Component-based software development (CBD) is based on building software systems from previously-existing software components. In CBD, reuse of common parts in component form can reduce the development cost of new systems, and reduce the maintenance cost associated with the support of these systems. However, existing programs have usually been built using another paradigm, such as the object-oriented (OO) paradigm. OO programs cannot be reused rapidly or effectively in the CBD paradigm even if they contain reusable functions. In this paper, we propose a technique for extracting components from existing OO programs by our new refactoring "Extract Component". Our technique of refactoring can identify and extract reusable components composed of classes from OO programs, and modify the surrounding parts of extracted components in original programs. We have developed a system that performs our refactoring automatically and extracts JavaBeans components from Java programs. As a result of evaluation experiments, it is found that our system is useful for extracting reusable components along with usage examples from Java programs.

Keywords: Component-based development (CBD); Refactoring; Object-oriented programming; Software reuse; Software component; JavaBeans

* Corresponding author.
E-mail addresses: washizaki@acm.org (H. Washizaki), fukazawa@waseda.jp (Y. Fukazawa).

0167-6423/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2004.11.007

1. Introduction

Component-based software development (CBD) has become widely accepted as a cost-effective approach to software development [35]. In CBD, software development is considered to involve the composition of various software components. CBD is capable of reducing developmental costs and improving the reliability of an entire system.

In this paper, we use object-oriented (OO) programming language for the implementation of components. CBD does not always have to be object-oriented; however, it has been indicated that using OO paradigm/language is a natural way to model and implement components [13]. In fact, some of the practical component architectures, such as JavaBeans [12] and Enterprise JavaBeans (EJB) [5], are based on OO technologies.

In CBD, the reuse of common parts in component form can reduce the development cost of new systems, and reduce the maintenance cost associated with the support of these systems. However, not all components corresponding to functional requirements are already available in all possible contexts. In contrast, there are many program repositories available on the Internet such as SourceForge.net [34]. At these on-line repositories, programmers can obtain a large amount of OO program source codes and binary codes. There is a possibility that programs, which partially fulfill the required functionalities, exist among these available OO program source codes. If such parts of existing OO programs could be easily reused as components, programmers could develop software by means of CBD by utilizing these programs.

However, since OO classes usually have complex mutual dependencies, it is difficult to reuse parts of existing OO programs composed of classes rapidly and effectively. If a significant function is realized by a set of classes, programmers who want to reuse the function must examine the dependencies among related classes and acquire all depending classes. Since such manual examination activities entail a high cost for programmers, the merits of reuse might be reduced or lost. Therefore, it is necessary to transform a part of an existing OO program into a component that has no dependence on elements outside itself. However, current CBD methodologies mostly lack a systematic decomposition algorithm [33].

Moreover, even if the components can be extracted from existing programs, it is difficult to identify the appropriate use of the extracted components only by referring to the source codes or public interfaces of the components. Therefore, it is preferable to acquire a usage example along with the extracted components.

In this paper, we propose a technique for identifying structurally reusable candidate parts of OO programs according to our definition of the reusable component based on JavaBeans [12], and transforming these parts into reusable components automatically by our new refactoring, “Extract Component”. Our technique targets Java language as the OO programming language and JavaBeans as the fundamental component architecture. Our technique accepts any kind of Java programs as the extraction target whether these programs use specific coding standards or structural templates. Moreover, we show that our extraction technique is useful for acquiring usage examples for the extracted components.

In the following, we first define a class relation graph (CRG) that represents the relations among classes/interfaces in the target Java program. Next, using a CRG, we propose a technique for extracting components from OO programs, and changing the