A Conceptual Framework for Identifying Business Components

Hangjung Zo
University of Wisconsin-Milwaukee

Hemant Jain
University of Wisconsin-Milwaukee

Follow this and additional works at: http://aisel.aisnet.org/amcis2001

Recommended Citation
http://aisel.aisnet.org/amcis2001/246

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2001 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
A CONCEPTUAL FRAMEWORK FOR IDENTIFYING BUSINESS COMPONENTS

Hangjung Zo  
University of Wisconsin at Milwaukee  
hangjzo@uwm.edu

Hemant K. Jain  
TCS Professor  
University of Wisconsin at Milwaukee  
jain@uwm.edu

Abstract

Component-Based Software Development (CBSD) has been considered as one of the most effective solutions to overcome the current “software crisis” (Brown and Wallnau 1998). This paper focuses on the issue of designing components, and proposes the framework for identifying components. Two main characteristics of the framework – practicability and flexibility – are explored. The proposed framework is based on the domain analysis through Unified Modeling Language (UML).

Keywords: Software reuse, CBSD, component identification, clustering, UML

Introduction

Even though CBSD includes all aspects and phases of the software development lifecycle, generally, two issues of CBSD are considered to be most critical (Herzum and Sims 2000). One is the creation and deployment of systems assembled from components, the other is the development and harvesting of such components (Kang 1999, Krutchen 1998). This paper addresses the second issue of developing components, and proposes a conceptual framework for identifying components. Based on the domain model represented in UML, this framework will help identify components during the design phase.

The proposed framework in this paper is just one part of the project aimed at developing new CBSD methodologies, component design tools, and component repositories. The main characteristic of this framework is practicality, because we need to apply this approach to real projects for component development. Another unique point of this framework is that we allow interventions of users (domain analysts, component designers, component assembler, etc.) as much as they can during the identification of components. Since users, such as domain analysts, have a lot of knowledge about a specific domain, we want to incorporate it with the component identification process. This feature of the framework gives users more flexibility to design components that satisfy their needs.

Conceptual Framework

Figure 1 shows the component identification process of the component design framework proposed in this paper. Given the results of domain analysis represented in UML (class diagram and use cases are mainly used as a starting point), the strength of associations between the classes representing the domain is identified. A matrix representing these relationships is generated. The classes are grouped into components using a hierarchical clustering method (Johnson and Wichern 1998, p. 738). The results of clustering are evaluated to access the quality of component design. The component designer can repeat the process until an acceptable design is obtained.
**Identifying the Strength of Association between Classes**

To measure strength of association between classes, coupling has been widely used (Briand et al. 1999, Chiambere and Kemerer 1994, Cho et al. 1998, Lorenz and Kidd 1994, Vitharana 2000). Chidamber and Kemerer defined coupling as “two classes are coupled when methods of one class use methods or instance variables defined by the other class” (Chiambere and Kemerer 1994, p. 486). Coupling has been measured by CBO, MPC, RFC, etc. (Briand et al. 1999, Yacoub et al. 1999). Another measure for strength of association between classes can be the number of times a class is reused (Lorenz and Kidd 1994). The more a class is reused, the more important it is.

In our framework, we combine coupling and the number of times classes are reused. All information we use in this stage is from domain analysis, specifically form use case and class diagrams. Use case diagrams show the system’s use cases and which actors interact with them, whereas class diagrams present the existence of classes and their relationships in the logical view of a system. From these diagrams, we can capture information about association and the number of times classes are reused in different use cases.

---

Figure 1. Conceptual Framework for Identifying Components

---

1CBO (Coupling Between Objects): the count of the number of classes to which it is coupled.
2MPC (Message Passing Coupling): the count of the number of send statements that is found in methods of one class to other classes.
3RFC (Response For Class): a measure of the number of methods that can potentially be executed in response to a message received by an object of that class.
Modified CBO (Coupling Between Objects) is employed to identify the strength of association between classes in our framework. MPC and RFC can be used as a coupling measure, also. Modified CBO is defined as follows:

**Definition: Modified CBO (MCBO)**

\[
\text{MCBO}(C_i, C_j) = \sum_k W_{ij} A(C_i, C_j) I_k .
\]

For \( \forall i, j, k, \) if \( C_i, C_j \in U_k (i \neq j) \)

\[
\text{MCBO}(C_i, C_j) = W_{ij} A(C_i, C_j) I_k.
\]

For \( \forall i, j, k, h, \) if \( C_i \in U_k, C_j \in U_h (i \neq j, k \neq h) \)

\[
\text{MCBO}(C_i, C_j) = \sum_{i,j,k,h} W_{ij} A(C_i, C_j) I_k.
\]

Where,

- \( C_i, C_j \) = Classes in the domain \( \Omega \)
- \( U_k, U_h \) = Use Cases in the domain \( W \)
- \( A(C_i, C_j) \) = \[ \begin{cases} 1 & \text{If there is an association between } C_i \text{ and } C_j \\ 0 & \text{Otherwise} \end{cases} \)
- \( W_{ij} \) = the weighted value of \( A(C_i, C_j) \)
- \( I_k \) = the importance value (weight) of use case \( U_k, (I_k > 1) \)

**Identifying Components**

After identifying the strength of association, classes are grouped into components using the hierarchical clustering method (Johnson and Wichern 1998). Hierarchical clustering method starts with the individual objects. The closest objects (having the strongest association) are first grouped; the initial groups are merged again according to their strength of associations until a cut off point is reached. The following steps are used in the hierarchical clustering algorithm for grouping \( N \) classes (Johnson and Wichern 1998, p. 740).

**Step 1.** Start with \( N \) clusters, each containing a single object class and an \( N \times N \) symmetric matrix representing strengths of associations between classes \( S = \{S_j\} \).

**Step 2.** Search the strength matrix for the strongest pair of clusters. Let us assume that \( U \) and \( V \) has the strongest relationship \( S_{UV} \).

**Step 3.** Merge \( U \) and \( V \). Label the newly formed cluster \((U V)\). Update the entries in the strength matrix by using \( S_{UAV} = S_{UV} + S_{VP} \). Delete the rows and columns corresponding to clusters \( U \) and \( V \) and add a row and column giving the strengths between \((U V)\) and the remaining clusters.

**Step 4.** Repeat step 2 and 3 until the cut-off point is reached.

**Evaluating Components**

Vitharana (2000) identified five managerial goals (cost effectiveness, ease of assembly, customization, reusability, and maintainability) and five technical features (coupling, cohesion, number of components, size of component, and complexity) that are closely related to managerial goals. He identified the strength of the relationship between technical features and managerial goals from a survey. In this paper, we adopt Vitharana’s model to evaluate the components identified above.

**User Inputs**

In order to allow flexibility to the designer, our framework allows component users’ intervention at every stage (Figure 1). Users can edit the result of the clustering process after they get an initial solution. Also, users can change the weights assigned to the use cases and input parameters of the clustering process. By changing user inputs, the designer can create alternative solutions.
The evaluation model of Vitharana (2000) enables users to compare alternative solutions. Once users get an appropriate and acceptable solution, they can proceed to the next phase – component design.

**Conclusion**

This paper proposes a new approach for identifying business components. Compared to previous methodologies, we expect that our framework is more flexible and practical. Even though we could not provide the optimal solution of component identification, we allow the intervention of users’ knowledge and experience to develop components. Therefore, by using this framework, component users can identify components from the domain model.

**References**

References are available upon request from the authors.