A Tool to Automate Component Clustering and Identification[1]

Soo Ho Chang, Man Jib Han, and Soo Dong Kim

Department of Computer Science,
Soongsil University,
1-1 Sangdo-Dong, Dongjak-Ku, Seoul, Korea 156-743
{shchang, mjhan}@otlab.ssu.ac.kr
sdkim@comp.ssu.ac.kr

Abstract. It is a key activity in CBD to identify high-quality components which have high cohesion and low coupling. However, component clustering is carried out in manual fashion by developers, resulting excessive time consumption and generating errors. In this article, we present an implementation of a tool which automates a component clustering and identification method. We show how we realize a clustering method as a tool and explain techniques applied in the implementation. Using the tool, component identification can be automated, and one can generate and navigate multiple configurations to find the most appropriate one for the project effortlessly.

1 Introduction

Component-based development (CBD) has been widely accepted as one of the representative reuse development paradigm. A component is a basic unit for reuse and provides a relatively coarse-grained functionality. A component typically consists of several related objects which collaborate to carry out system operations. Hence it is a key activity in CBD to identify high-quality components which have high cohesion and low coupling[1][2].

Several methods for identifying components have been proposed, but the manual application of the methods by developers is time-consuming and prone to generate errors. Hence, it is desirable to have tools to automate the process of the methods.

In this article, we present an implementation of a tool which automates a component clustering and identification method. We show how we modified the method for the purpose of realizing in a tool and key techniques applied in the implementation.

2 A Method for Clustering Component

The method that we chose for our implementation is known as one of the systematic methods with guidelines [3], and it consists of four steps as in figure 1. This method

[1] This work was supported by Korea Research Foundation Grant. (KRF-2004-005-D00172)

M. Cerioli (Ed.): FASE 2005, LNCS 3442, pp. 141 – 144, 2005.

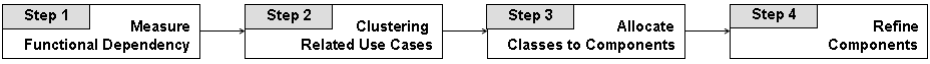

Fig. 1. The Overall Process

assumes that the fundamental artifacts of object-oriented modeling such as use case model, object model and dynamic model are available.

In this method, three types of relationships are considered for identifying components. In steps 1 and 2, functional dependency between use cases is used as the fundamental means to cluster related functions. The dependencies are measured with four criteria in step1 and related use cases are clustered in step 2. In step 3, functionality-to-data relationships expressed in a dynamic model such as sequence diagram are taken to assign related classes to candidate components. In step 4, dependency or coupling between classes is used to verify and refine the identified components. If there are two closely related classes which are separated into two components, it is identified and refined in this step

3 Implementation of the Tool

We develop the tool on Visual Studio .NET framework using C# language and MS Office Access. Hence, the tool runs on .NET framework with Office Access database installed. We applied a simplified version of object-oriented method. The tool has four modules, shown with «subsystem» stereotype in Figure 2; *Initializer*, *Configurator*, *Custerer*, and *Navigator*.

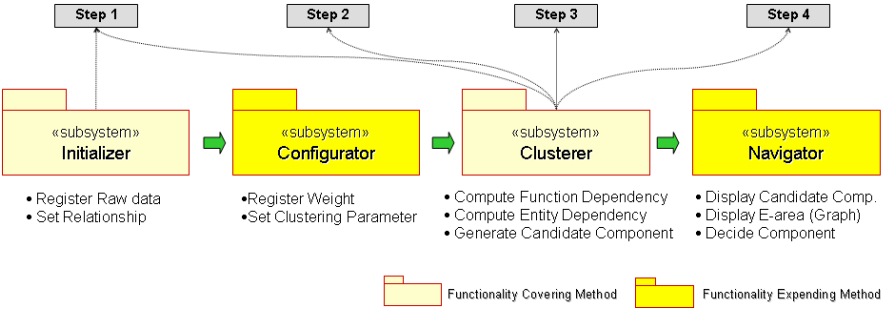

Fig. 2. Method and Subsystems of Tool

Initializer and *Clusterer* directly cover steps of the method, and *Configurator* and *Navigator* support additional functionalities for clustering in various conditions.

The *Initializer* module implements the step 1 of the method. It is to gather raw data needed for clustering; actors, use cases, classes, and to set the relationships among them. The *Configurator* module extends the step 1 with a functionality to register weight values for the metrics computing functional dependency (FD) and entity dependency (ED) and to set various parameters for resolving clustering conflicts.

The *Clusterer* module is to compute FD and ED, to cluster use cases into candidate components, and to assign classes into the components. Conflict occurrence during

automatic clustering is a common problem, and we use the following algorithm to resolve the conflicts mechanically.

```
…
// Classify conflict use case and duplicated use cases
For all use cases UC_i
   Search conflict components for a use case.
      If the number of conflict components >=
                                        numOfConf then
         conflictUC.usecaseList = UC_j
         conflictUC.conflictComponentsList =
                                  conflict components.

         add conflictUCList(conflictUC)
      Else if the number of conflict components
                                        > 1 then
         duplicatedUC.usecase = UC_j
         duplicatedUC.conflictComponentsList
                                  = conflict components.
            add duplicatedUCList(duplicatedUC).
      End if
Loop
// Discard subset of conflict component list.
..
// Make new component from conflict component list
.
..
// Assign duplicated use case
For i = 1 to duplicatedUCList.cnt
   // Form resolving use case conflict
   // for In-house Component
   AssignConflictUsecase( duplicateUCList[i].usecase,
         duplicateUCList[i].candiateComponentList)
   Loop
```

The *Navigator* module is to display multiple configurations of component set, and to let the user browse them and select the most appropriate configuration. The module computes and displays various measures about multiple component sets, so that users can made more logical decisions based on these measures.

4 Case Study with Rental Management

The rental management system is for managing rental operations in various rental-related businesses such as library, movie rental, and car rental. There is a good degree of commonality among these applications, and we were able to identify several components for this commonality.Our object-oriented analysis model only for entity layer includes 33 use cases and 6 entity classes. Figure 3 is a snapshot of *Initializer* showing the raw data of these use cases and classes entered.

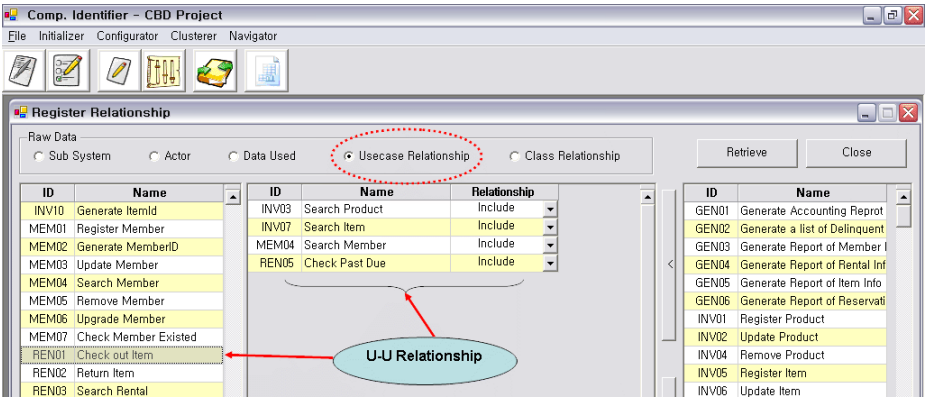

Fig. 3. Initialization

After *Configurator* gets parameters for resolving conflicts, *Cluster* computes automatically FD and ED and derives candidate components, as shown on the left-hand side of figure 4. The tool generates multiple configurations of component sets. Finally, the Navigator plots *economic area graph*s as shown in figure 4. It shows two relationships; one between the number of components and the value of 't' threshold, and the other between the granularity of components and the value of 't'. Using this information, user chooses a component set that best satisfies the requirement.

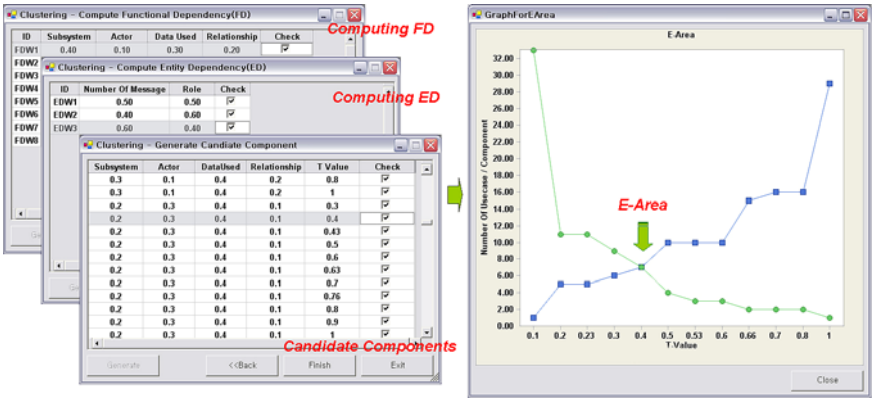

Fig. 4. Computation by *Clusterer* and Analysis by *Navigator*

5 Concluding Remarks

It is an essential activity in CBD to identify high-quality components which have high cohesion and low coupling. However, component clustering is carried out in manual fashion by developers, resulting excessive time consumption and generating errors. In this article, we presented an implementation of a tool which automates a component clustering and identification method. We showed how we realized a clustering method as a tool and explained techniques applied in the implementation. Using the tool, component identification can be automated, and one can generate and navigate multiple configurations to find the most appropriate one for the project effortlessly.

References

[1] Cho, Eun Sook, Kim, Soo Dong, and, Rhew, Sung Yul, "Domain Analysis and Preliminary Design Technique for Component Development", *International Journal of Software Engineering and Knowledge Engineering*, Vol. 14, No. 2, p.221-254, World Scientific Publishing Co., May 2004

[2] Choi, S., Chang, S., and Kim, S., "A Systematic Methodology for Developing Component Frameworks," Lecture Notes in Computer Science 2984, Proceedings of the 7th Fundamental Approaches to Software Engineering Conference, 2004

[3] Kim, S., Chang, S., "A Systematic Method to Identify Software Components," Proceedings of APSEC 2004, Nov. 2004.