Towards a Set of Application Independent Clustering Criteria within an 
Architecture Recovery Approach

Aline Vasconcelos¹,²
²CEFET Campos
Dr. Siqueira, 273 – Pq. Dom Bosco – ZIP
28030-130 - Campos dos Goytacazes - RJ
aline@cos.ufrj.br

Cláudia Werner¹
¹COPPE/UFRJ
P.O. Box 68511 – ZIP 21945-970 – Rio de Janeiro – RJ – Brazil
werner@cos.ufrj.br

Abstract

Architecture recovery is usually supported by a set of tools that implement techniques, such as concept analysis, clustering, data mining and pattern matching for abstracting low-level source code elements into higher-level architectural elements. However, the decision making in this process usually depends on a previously defined high-level model of the system architecture or on criteria that are application dependent. In this paper we present a set of application independent clustering criteria to support architectural elements definition. These criteria help the user in his decision making and can be reused in many different contexts.

1. Introduction

Many approaches to architecture reconstruction have already been proposed [3] [4] [5]. Legacy systems in general do not have an up-to-date architectural documentation that can support system maintenance, reengineering, reuse etc. Therefore architecture recovery is needed.

The recovery process is a three-step process, involving the extraction of information from system available sources (e.g. source code, documentation, execution traces, system experts), abstraction of this information to an architectural level, and presentation of the abstracted information. Legacy systems are often very large and the abstraction phase is a challenging one, requiring appropriate criteria to architectural elements definition. Moreover, the presentation phase tends to be as challenging as the abstraction level, since a large amount of information needs to be visualized.

2. The architecture recovery process

The proposed process is depicted in figure 1.

Figure 1. Architecture recovery process.

It starts by the static reverse engineering and use-case modeling activities. The static reverse engineering aims at the extraction of a UML class model from Java source code. This model is where source code elements are clustered into architectural elements. The use-case modeling determines execution scenarios to guide the dynamic reverse engineering and to establish the mapping from system
functionalities to source code. The dynamic reverse engineering aims at the collection of system execution traces and the extraction of UML sequence diagrams. Architectural elements generated in the static view are mapped to these diagrams, allowing the description of the control and data flow in the architecture.

3. Clustering criteria

The criteria to architectural elements definition (i.e. clustering source code elements into architectural elements) are mainly based on the application of data mining techniques over the collected execution traces. The goal of using data mining is to detect similarities among execution traces, revealing source code elements that perform common system functionalities. These source code elements must be grouped into an architectural element. To this end, two data mining techniques are employed: sequential pattern mining [2], and association rules mining [1].

The sequential pattern mining aims at the detection of interaction patterns, which represent sets of methods that are frequently invoked during system execution. They characterize a common behavior in the system. The goal of the association rules mining is to detect strongly related classes that are common to a set of use-case realizations.

For the first form of mining, use-case scenarios (i.e. execution traces) are mapped to sequences of client transactions in a database and method executions are mapped to transactions containing only one item. The support for an interaction pattern is the number of use-case scenarios in which it appears. If it appears in a number of scenarios greater than a minimum support threshold value, than its classes are suggested as candidates to be clustered.

For the association rules mining, use-case scenarios are mapped to database transactions and class occurrences to data items. Classes that appear together in a number of use-case scenarios greater than a minimum support threshold value and that have a confidence relation among them greater than a minimum confidence threshold value are suggested as candidates to be clustered. Confidence c means the percentage of transactions containing one item A in which another item B appears. In this case we say that A => B with confidence of c%.

However, not all the classes will be suggested as candidates to be clustered based on the results of the data mining. This happens due to intersections in interaction patterns or due to difficulties in interpreting the resultant association rules. In this case static analysis can be employed. The static analysis in this work involves looking for classes with similar names that are candidates to be clustered and grouping classes into architectural elements to which they have a high coupling value.

4. Conclusions

Most of the approaches to architecture recovery are based on one form of system analysis or propose criteria that are application dependent. This paper presented a set of application independent clustering criteria to architectural elements definition based on dynamic and static analysis. The criteria are supported by a tool set integrated to the Odyssey environment [6], but the process is semi-automated since domain knowledge is needed. The goal is to reduce the required effort in the recovery process by providing the user with some guidelines and tool support. The approach and criteria can be reused in many different contexts.

We have already experimented the approach in two systems, obtaining satisfactory results. We are now working with larger sets of execution traces in order to refine the criteria and verify the scalability of the approach. As future work we intend to use some architectural analysis method in order to verify quality attributes satisfied by the recovered architecture.

5. References


