Nebras Classifier: A Generic Multi-Domain Reusable Component

Seyyed Vahid Hashemian
Nebras Informatics Co.
Hashemian@nebrasinfo.com

Sasan Dashtinezhad
Nebras Informatics Co.
sasan_d@nebrasinfo.com

Navid Khosravi
Nebras Informatics Co.
CEO@nebrasinfo.com

ABSTRACT

Nebras Classifier is a distributed model-view component developed to perform one type of information handling –namely classification- using both COM and CORBA communication facilities. In this component, model has a hierarchical structure capable of storing multiple instances of domain CORBA objects in its nodes and has facilities to perform special queries on this structure. The view is a graphical representation for this model. A third party named Semantic Engine that is domain dependent, may be used in interaction with these two parts using proxy design pattern to perform required feasibility checks or domain dependent actions on the classifier.

Categories and Subject Descriptors
Analysis, Design & Architecture, Programming, , Patterns.

General Terms
Management, Design

Keywords
BOM, Classifier, CORBA, COM, Reusability, Pattern

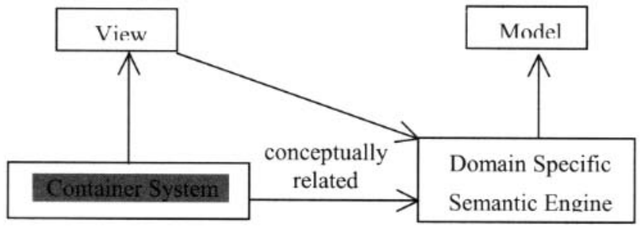
1. INTRODUCTION

One of the powerful concepts that help in organization and management of information is classification. This concept is applicable wherever we deal with some pieces of information. In the field of computer software, classification is best fitted into Information Systems, in which, there are lots of different objects or entities.

Nebras Classifier is a reusable component developed to perform classification of CORBA objects in object-oriented systems regardless of object types. Also, it can be embedded into almost all object oriented software systems. In order to be classifiable, every class of object must support a specific interface exhibited by this component.

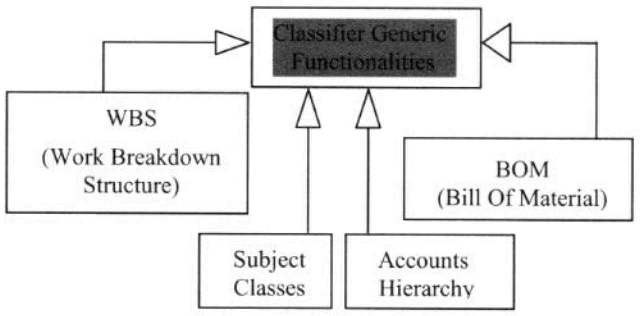
2. STRUCTURE OF NEBRAS CLASSIFIER

Three constituent parts of classifier component interact with each other as shown in the following diagram:



The main two parts of Nebras Classifier are domain independent and implement the core semantics common to all hierarchical structures shown below. If needed, a domain specific Semantic Engine must be developed from the container system's point of view, however it must support special interfaces to be interacted by the view. The view instances are always consistent with the model and are reflected by the changes done to the model.

Examples of the container system are displayed in the following diagram:


Classifier Generic Functionalities
WBS (Work Breakdown Structure)
BOM (Bill Of Material)
Subject Classes
Accounts Hierarchy

As far as Nebras Classifier would be used in distributed environments, it is possible that different users open the same classifier on different machines. Therefore some policies are necessary for change propagation and access control among classifiers, such as developing new components playing the role of manager for change propagation and access control that run on the server side. These components are accessed when necessary by the view that is running on the client side. Communication between these two sides (server and client) is done using COM+. Variations of Publisher-subscriber [1] and Observer [2] patterns are used for change propagation.

In addition to publisher-subscriber pattern, some other patterns used in development of Nebras Classifier are:

- Layers [1]
- Model-View-Controller [1]
- Proxy [2]

The generic functionalities, which are commonly required in all of the above industrial components, and the applicability of the above patterns in Nebras Classifier, will be presented in the demonstration.

In this demonstration we are going to present:

1. How such a component should be extended to support domain dependent behavior.
2. How change propagation is resolved in a distributed objects environment.

How simultaneous access to shared models are controlled when viewed by multiple views.

4. Which famous structural and behavioral patterns we used in the development of Nebras Classifier.
5. How such a generic component plays a great role in most of the famous domain dependent hierarchical structures such as BOM, WBS, accounts hierarchy, and classification systems by applying extension mechanisms.

3. ACKNOWLEDGMENTS

Our thanks to all Nebras Informatics Co. Staff, specially to Ahmad Zarei, Mohammad Bakuii, Arash Farzin, Hadi Salimi, Amir Rasekh and Reza Sadoddin.

4. REFERENCES

[1] Buschmann, F., R. Meunier, H. Rohnert, et al. *Pattern-Oriented Software Architecture—A System of Patterns.* Wiley and Sons Ltd., Chichester, England, 1996.

[2] Gamma, E., R. Helm, R. Johnson, et al. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA, 1995.