HyperModel – A Distributed Laboratory for Teaching of Object Oriented Modeling

Nenad MAROVAC
Department of Computer Science
San Diego State University
San Diego, CA 92182
nenad@sdsu.edu
www.multimedia.sdsu.edu

KEYWORDS: *Teaching, Software, Modeling, Distributed*

ABSTRACT: *Understanding and learning principles of Object-Oriented Modeling is a long and tedious process. This makes adopting the OO modeling in software design more difficult. In this paper we present a system used to classify, describe and store OO models and associated artifacts into a world-wide distributed platform. This enables students anywhere in the world to access stored models and associated artifacts that can assist them in building desired software systems.*

1 INTRODUCTION

At San Diego State University in the Department of Computer Science we have been teaching Object-Oriented Modeling and design for over 10 years. During this process we quickly became aware that students take a very long time to understand and learn the principles of OO modeling and to become confident that the model they produced properly reflected the philosophical, constitutive and semantical properties of the given problems. This is also evident in the difficulty the students then have in using the produced models to create logical and detailed design of software systems for solving the given problems.

Although there are many books and articles written on the topic of Object Oriented Modeling and Design and UML [2, 3, 4, 5, 6, 12, 14, 15], students find them difficult to read and of limited use in helping them to master the problem. Two favorite books of ours are by Rumbaugh [13] and Larman [7]. Rumbaugh provides sound philosophical reasoning for using Object Oriented Modeling as well as very good introduction to Object Oriented Analysis and Modeling. He uses OMT language and methodology, which is not necessarily a big drawback as a significant part of UML is directly based on OMT language. However, the Rumbaugh book does have a considerable weakness in progressing from modeling in the analysis task to creating models for design. Furthermore, it does not address iterative software construction. The book by Larman lacks the philosophical reasoning. However, it is a very good practitioner's book in that it guides students through different diagrams (models) in UML repertoire; and it suggests when and how to use them in problem analysis, as well as in software design and construction phases.

The discussion presented above identified issues that we felt needed addressing in order to successfully teach Object Oriented Modeling in the problem analysis stage and in the Object Oriented software design and construction.

2 HYPERMODEL

The essence of our project was to build a world-wide distributed platform for storing Object-Oriented models and associated artifacts, such as documents and implementation code. The models include models obtained from the Object Oriented analysis of the problem; design models obtained by producing the logical design of the solution; and sometimes detailed implementation models. All these models can be created using any modeling tools, such as Rational Rose [14].
The analysis models comprise:

1. The Analysis Object Model [13] or the Conceptual model in UML parlance [7] shows the objects and concepts within the problem, their properties, and the relationships between them.

2. The Sequential Diagrams and State Diagrams [7] illustrate the dynamics within the problem.

The design and implementation models include:

1. The Class Model obtained by expanding the Conceptual Model by adding concepts and objects identified in the logical design of the system.
2. Collaboration diagrams [10] and State Diagrams illustrating the dynamics within the logical solution to the problem.

The analysis models are used to classify the problem in our system. In order to do that the analysis models must be first translated into flagged sentences [8, 9]. The flagged sentences form a set of documentation statements that completely describe the conceptual, constitutive, topological and semantical properties of the problem and the associated models and software components.

The flagged sentences format of the models is used for two purposes:

1. To store the model into the HyperModel knowledge database and to classify the model. In fact the original analysis models are stored together with its flagged sentences representation. The properties encoded in the flagged sentences are used to classify the problem that is represented by the model. At that time or any time thereafter we can add artifacts associated to the model entity. The most important artifacts are: the problem statement, Use Cases describing the required functionality in the problem. Other artifacts include the design models, the pseudo-code, libraries of completed code that may implement portions of the model, documentation, etc. The design models and all other associated artifacts are stored in their original formats into a HyperNet distributed knowledge database.
2. To search for a match. When we are given a new problem to deal with, we can create a model of the given problem, translate the model into flagged sentences and then search the distributed database for models that are already stored in the HyperModel, and have a desired match quality. Candidate software models are selected from the knowledge database based on the degree of similarity between their descriptions contained in the flagged sentences for these models, and the problem description, again expressed in flagged sentences. The "similarity" is measured by a non negative value that expresses the amount of effort we have to invest in translating a candidate model into the model corresponding to the given problem [1].

3 HYPERMODEL IN SOFTWARE ENGINEERING EDUCATION

HyperModel is a very useful tool for teaching Object Oriented Modeling and Design. When we define a problem, a student creates either a segment or an entire Conceptual Model for the given problem. The student then searches for a similar model within the HyperModel. The conceptual model must be translated into flagged sentences and the system searches for all similarity candidates that are already expressed in flagged sentences format. These candidates are presented to the student in the descending order of the similarity. The student then compares his/hers model to the similarity candidates. By studying these models and analyzing the similarities and differences between the models the student can learn by identifying omissions and errors. Furthermore, if an acceptable match is found the student can extract all associated artifacts and use them to advance in education in software modeling, engineering and implementation.

4 HYPERMODEL IN SOFTWARE CONSTRUCTION

We start with the same steps. First, we construct a Conceptual Model for a given problem; the Conceptual Model is then presented to HyperModel in order to search for similarity candidate. When a suitable similar model is found, a software engineer decides whether the found model is adequate for adoption in full. If this is the case the engineer can extract associate artifacts and use them in constructing the required solution.

Very often the candidate models do not closely match the given problem. In that case the most similar model can be modified to completely reflect the problem at hand. These modifications will determine the required modifications to the artifacts associated with the found model in order to obtain the best design and implementation for the required solution. The modified model and artifacts are then also stored into HyperModel database for future use.

5 HYPERMODEL AND HYPERNET

The design and implementation of the infrastructure for HyperModel is based on HyperNet; it uses the distributed database as well as the store, classification, search and the retrieve mechanism of HyperNet [10, 11]. HyperNet is an authoring and browsing system facilitating creation and navigation of multimedia documents. It was conceived and designed to provide a mechanism for very fast choreographing and configurating of hypermedia documents. HyperNet very closely resembles WWW in that the system provides users with a hypermedia integration and browsing environment to document component databases distributed across the world, provided they have access to the Internet network and are valid HyperNet sites. It supports HTML and its own language Scripts, and HTTP as well as its own protocol called HNET. Its strongest feature is the document component classification mechanism. Each component can be classified on a number of properties including keywords. The search for components using these properties is completely transparent, i.e. in a search request, there is no need to specify a path (either absolute or relative); just properties of the component and one, more or all sites to be searched.

CONCLUSION

In this paper we presented a system that is used both as a tool for teaching students Object Oriented modeling and design, as well as a vehicle for software engineers to assist them in the design and construction of software to solve given problems. The first version of the system has been implemented as a thesis project [1]. The main work for the future is to find a better algorithm to identify the similar models to a given problem; to construct an efficient technique to semi-automatically determine the differences between the model corresponding to the given problem and the similar models; and lastly to project these differences to the artifacts associated with the similar models. This would greatly reduce the time required to identify required modifications to the associated artifacts to obtain complete solution to the given problem.

REFERENCES

[1] BENOTHMANE M. Hyper Solver Expert System, san Diego: *A thesis presented to the Department of Computer Science at San Diego State University*, 2001.

[2] BOOCH G., RUMBAUGH J., & JACOBSON I. *The Unified Modeling Language User Guide.* Addison Wesley, 1998.

[3] DE CHAMPEAUX D. O-O Development Process and Metrics. Prentice Hall, 1997.

[4] FOWLER M. *UML Distilled.* Addison-Wesley, 2000.

[5] JACOBSON I., BOOCH G., & RUMBAUGH J. *The Unified Software Development Process.* Addison Wesley, 1999.

[6] JACOBSON I. *Object-Oriented Software Engineering.* Addison Wesley, 1992.

[7] LARMAN C. *Applying UML and Patterns Prentice-Hall.* 2002.

[8] MAROVAC N. *Guidelines for Embedded Software Documentation.* ACM Software Engineering Notes, Vol. 19, N° 2, April 1994.

[9] MAROVAC N. *Embedded Documentation for Semi-automatic Program Construction and Software Reuse.* ACM Software Engineering Notes, September 1997.

[10] MAROVAC N. & OSBURN L. *HyperNet - A tool to choreograph world wide distributed hypermedia documents.* Computers & Graphics, Vol. 16, No. 2, Pergamon Press 1992.

[11] MAROVAC N. *Link associated computations in HyperNet.* The British Computer Society: The Computer Journal, Vol. 37, No. 2.

[12] RUMBAUGH J., JACOBSON I., & BOOCH G. *The Unified Modeling Language Reference Manual*. Addison Wesley, 1999.

[13] RUMBAUGH J., et al. *Object-Oriented Modeling and Design. Prentice-Hall*, 1991.

[14] QUATRANI T. *Visual Modeling with Rational Rose 2002 and UML*. Addison Wesley, 2003.

[15] TEXEL P., & WILLIAMS C.B. *Use Cases Combined with Booch, OMT, UML.* Prentice-Hall, 1997.