Framework for Building Ontology-Based Dynamic Applications

Samuil Nikolov, Anatoliy Antonov

Abstract: *The publication describes the main principles of operation of a commercial software framework that is used for developing financial applications represented with ontologies. It is shown that the ontology structure can be extended and modified, without disturbing the working process of the application. The framework generates dynamic user interface used to define class instances. It effectively hides the theoretical background of the product and allows the end user to work with terms familiar to him.*

Keywords: *Ontology, Dynamic Ontology Construction, Knowledge Representation, Applications with Dynamic User Interface.*

INTRODUCTION

The goal of the publication is to describe the principles of operation of a flexible and extendable framework, developed by the authors, that allows real time configuration of ontology, processing and analyzing the data in it. The main advantages of the proposed solution are:

- The ontology representation is hidden for the end user;
- Possibility for the ontology specialist to extend it without interrupting the working process.

Most existing solutions for creating and analyzing ontologies deal with the problem of semantic processing of knowledge in internet. The web ontology language OWL [1] is most widely used and a lot of products are based on it. Most often, analysis is performed on the entire ontology or between different ontologies. Chimaera [6][8] supports merging multiple ontologies together and diagnosing individual or multiple ontologies. PROMPT [5] manages multiple ontologies in Protégé. OntoBroker [4] analyzes ontologies defined with OntoStudio [3]. OntoMerge [9] creates translations by merging two or more ontologies. DYNAMO [7] supports an ontologist in the process of extracting ontologies from free defined text. From the reviewed solutions, Protégé [2] is the most complete, supported and used framework for building and analysis of ontologies [18][19][20]. The means for describing them include:

- Definition of classes and attributes;
- Definition of links between the classes.

The means for analysis include:

- Defining services for domain processing;
- Defining user interface for creating instances and specifying their attributes;
- Persisting class instances and their attributes.

The result generated in Protégé is a static ontology definition [10] that can be analyzed by the end user. However, he has to be a specialist in the field of ontologies and to understand their internal organization, to be able to analyze it [11].

In the presented solution the end user can add class instances in real time without having any knowledge of ontologies. The specialist in ontologies and in the specific domain which is modeled prepares for the user the following:

- Classes and links between them;
- User interface for attribute input and business logic for validation of the entered values;
- Business logic for analyzing the class instances created by the user.

In real time, the end user can:

- Create new instances of the defined classes;
- Edit the attributes of the instances and store them in a persistent medium;

- Obtain the results of analysis over the instances added by him. The analysis is incorporated in the business logic defined by the domain specialist;
- Add new classes to the ontology without interrupting the work process. The new classes are provided by the domain specialist.

1. Structure of the proposed solution

1.1. Applications in the Risk Framework environment

The dynamic ontology construction will be demonstrated in the commercial product Risk Framework [14][15][16], developed by the authors. It is used for creating series of financial applications, which are essentially ontology domains.

Every application developed for the environment uses a set of classes represented in the system by models. The product is based on a Clips interpreter [13] and C++ interface layer. The layer provides:

- Ability to add and delete class instances in real time;
- Dynamically generated user interface for editing attributes of the created instances;
- Persisting and loading ontological data;
- Management of the set of models, defining the domain;
- Starting the execution of business logic incorporated in the models.

The Clips interpreter processes the models, which includes:

- Loading of models that represent the ontology classes;
- Communicating the dynamic user interface description in the models to the C++ layer for rendering;
- Executing Clips rules representing the business logic and calculating proper values for some of the attributes of the current class.

1.2. Models

The models in Risk Framework encapsulate the ontology elements Class, Properties and Forms, used in [2]. The different modeling approach used by the authors corresponds to classes in Object Oriented Programming [17], which unite the data with the operations performed on them. The models are created by a specialist in the domain represented by the specific Risk Framework application. Every model is a Clips program which is interpreted in real time. It contains:

- Facts about user interface definition [12];
- Validation logic for the user input;
- Logic, defining the connections between the current class and the other classes in the domain;
- Business logic for analyzing the data input by the user and/or contained in persisted instances of other classes.

The interface layer of the system controls one instance of the Clips interpreter, where a single model can be loaded as a result of user actions or due to business logic defined in other models. Without interrupting the process, the ontology specialist can replace all the models except the currently loaded one and add new ones. Thus the structure and contents of the modeled ontology can be changed in real time. In case a new class is needed, the domain specialist creates a new model, describing the user interface for it and defining the business logic. Then he submits it to the end user, who can load it using the interface layer and create instances of the new class. The ontology definition limits the number of instances the user can create and persist.

1.3. Sessions

Sessions in Risk Framework are class instances persisted by the end user. Every session

is uniquely defined by an identifier, containing a string (Item ID) freely entered by the user. The session contains information about the class which instance it is, about the values of the attributes entered by the user and about the links created to persisted instances of other classes. They do not contain information about user interface description. This allows modifying the visualization of the persisted session attributes in real time by changing the dynamic user interface descriptions in the model.

2. Principles of operation of the Risk Framework environment

The principles of operation of Risk Framework will be described to show that the end user does not need to have any knowledge of ontologies and that the theoretical background of the system remains hidden for him. The C++ interface layer of Risk Framework has two main elements – a tree control and a dialog window.

The tree contains a list of identifiers of instances called Items in the system. It can be used to create, delete and modify class instances. Figure 1 shows an example of such list containing one identifier - RFW System and one class instance – session, associated with it. The session is from a class called “Catalogs in Risk Framework”.

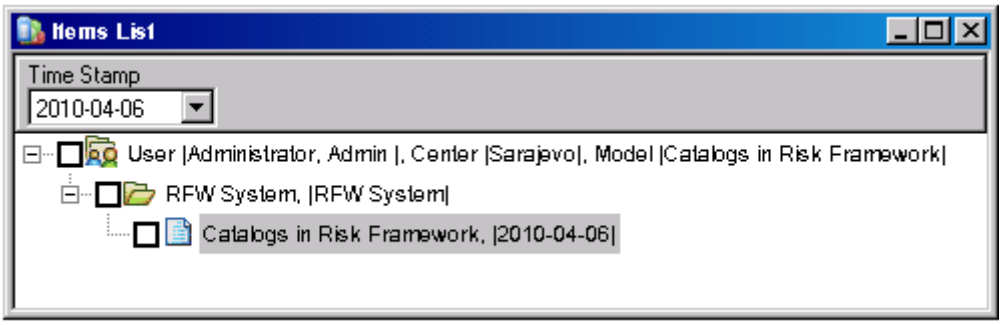

Figure 1. List of identifiers in a tree control

The other interface element – the dialog window is used for editing attributes of class instances and activating the business logic associated with them. It is opened by choosing and opening any of the sessions shown in the tree control. Besides class attributes, the dialog window can also contain tables with links to other class instances. The contents of the window are dynamically generated in accordance with the description in the model, currently loaded inside the interpreter. Figure 2 shows a part of the user interface, generated for the class Catalogs in Risk Framework.

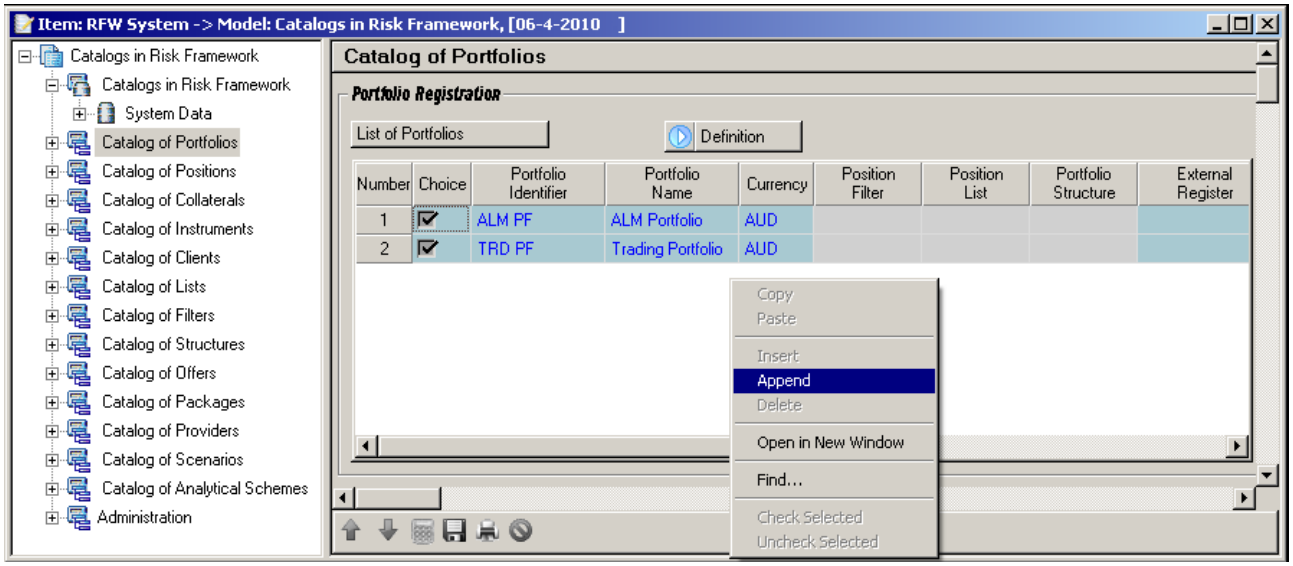

Figure 2. Dynamically generated user interface in the dialog window

The tree control in the left part of the dialog window is used as a navigation map. The right part is the dynamic interface. The one in the screenshot contains a group box - Portfolio Registration, two labels – List of portfolios and Definition, a control button with an image and a table with extendable contents. The buttons in the lower part of the dialog window can be used to start the business logic inside the model, store or print the data entered by the user. By adding rows in the table and entering identifiers in the proper column, the user can link the current catalog instance with concrete instances of another class in the ontology - Portfolio Definition, which in this case describes data about a financial portfolio.

3. Example of creating a Portfolio Management application

To develop the application, the portfolio management specialist, who is also a specialist in ontologies, creates Clips models each representing a class from the domain – portfolio, position, list, filter, instrument, client, yield curve. During the design the possible links between the classes are determined. For instance, the link between portfolio and position is modeled by adding an interface table in the portfolio model that will contain the item identifiers of the positions associated with it. Some of the created models are used as organizational models and aid the operation of the other ones. Such is the Catalogs in Risk Framework model shown in Figure 2.

When the user starts the Risk Framework environment, the identifier list shown on Figure 1 contains one default item identifier - RFW System, which has no associated session. The user has to load the organizational model Catalogs in Risk Framework from the interface layer and to create an instance of the class described by it. When the session is opened, the internal user interface definition is dynamically rendered (Figure 2).

The user interface defined by the domain specialist allows the user to add new rows to the tables in the catalog. The user can add rows to the “list of portfolios” table and input identifiers of portfolios that will be included in the catalog. In the example shown there are two identifiers added - ALM PF and TRD PF. To create actual instances of the class Portfolio Definition associated to the entered identifiers, the user has to activate the button Definition. Thus, he starts the business logic inside the model Catalogs in Risk Framework. Two main events are triggered:

- The Portfolio Definition model is loaded in the Clips interpreter;
- The C++ interface layer is activated, modifying the contents of the tree control. The default identifier RFW System is replaced by the two identifiers entered by the user in portfolio table (Figure 3).

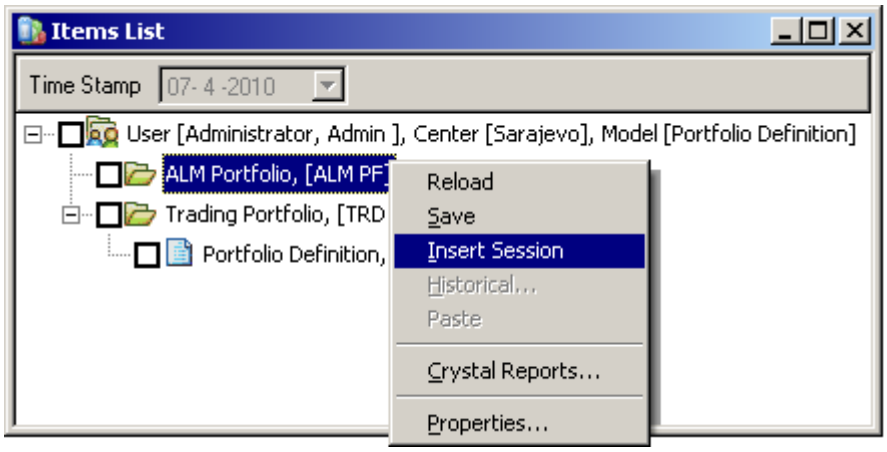

Figure 3. Contents of the tree control after activating the business logic in the model Catalogs in Risk Framework

By the described activation of the business logic a transition is made from defining

instances of the organizational model Catalogs in Risk Framework to defining instances of the class Portfolio Definition. From the end user point of view, defining the described ontology consists in adding portfolios and editing their parameters. The underlying theoretical model is completely concealed behind the described user interface. The ontological links between the classes are implemented by adding interface tables to the generated user interface and business logic that controls the validity of their contents. For example, the user will not be able to set up a backward link between a Portfolio Definition class instance and the Catalogs in Risk Framework class instance, because the user interface, generated for defining portfolios does not contain a table with catalogs. It does contain, however, a table for defining positions, which models in their turn contain tables for defining financial instruments etc.

In the new state of the tree control, the user can create new instances of the Portfolio Definition class using the drop-down menu provided by the interface layer. Figure 3 shows the state of the tree element with a newly created instance for the identifier TRD PF and the process of creating a session for the other identifier - ALM PF. The user can store sessions in the persistent medium, reload them if another user has modified them and generate reports from the data stored inside. The newly generated sessions can be opened which will dynamically render the user interface described in the model Portfolio Definition. In it, the user can input general portfolio attributes and the position identifiers that will be contained in the specific portfolio. By activating a similar Definition button, the C++ layer will load their corresponding model - Position Data and reload the tree control with the entered position identifiers.

All instances of the classes, defined by the specialist, are created in a similar way. Adding a new class to the ontology consists in creating the new model and in corresponding modification of all the models that have ontological link to it.

CONCLUSIONS AND FUTURE WORK

The proposed solution for ontology data definition is more flexible and extendable than the examined existing solutions. Moreover, no theoretical knowledge of ontologies is required from the end user. This is due to the data structure and principles of operation described in the publication. The dynamically generated user interface hides the theoretical backgrounds of Risk Framework environment. The limited size of the publication did not allow reviewing some other important aspects of the solution:

- Organization and operation of the persistent layer;
- Interaction between the C++ interface layer and the Clips interpreter;
- Mapping of stored session properties to models modified by the ontology specialist;
- The business logic implemented in models;
- Historical development of the persisted sessions allowing storage of different attributes for a class instance in different time points.

The described environment is used for executing series of financial applications like Basel II Risk Management [14], Operational Risk Management [15], and Asset and Liability Management [16].

The future development of the product is connected with developing a web solution based on the same principles of operation and developing applications outside the field of finance.

REFERENCES

[1] OWL. Web Ontology Language. W3C. - http://www.w3c.org/TR/owl-features/
[2] Protégé. - http://protege.stanford.edu/ontologies/ontologyOfScience.
[3] OntoStudio. - http://www.ontoprise.de/en/home/products/ontostudio/
[4] OntoBroker http://www.ontoprise.de/en/home/products/ontobroker/

[5] Noy N., Musen M. The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping, Stanford Medical Informatics, Stanford Univ., 2003.

[6] McGuinness, Deborah L., Richard Fikes, James Rice, and Steve Wilder. "An Environment for Merging and Testing Large Ontologies." Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000). Breckenridge, Colorado, USA. April 12-15, 2000.

[7] Zied Sellami, Marie Pierre Gleizes, Nathalie Aussenac-Gilles, Sylvain Rougemaille, Dynamic Ontology Co-construction based on Adaptive Multi-Agent Technology, KEOD 2009 - Proceedings of the International Conference on Know ledge Engineering and Ontology Development, Funchal - Madeira, Portugal, October 6-8, 2009.

[8] McGuinness, Deborah L., Richard Fikes, James Rice, and Steve Wilder. "The Chimaera Ontology Environment." Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000), Austin, Texas. July 30 - August 3, 2000.

[9] Dejing Dou, Drew McDermott, and Peishen Qi 2003 Ontology Translation on the Semantic Web. In Proc. Int'l Conf. on Ontologies, Databases and Applications of SEmantics (ODBASE2003), LNCS 2888, pp. 952-969.

[10] Hai H. Wang, Natasha Noy, Alan Rector, Mark Musen, Timothy Redmond, Daniel Rubin, Samson Tu, Tania Tudorache, Nick Drummond, Matthew Horridge, and Julian Sedenberg. Frames and OWL side by side. In 10th International Protege Conference,Budapest, Hungary, July 2007.

[11] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications Third International Semantic Web Conference, Hiroshima, Japan (2004), pp. 229-243.

[12] Paskalev Pl., Nikolov Vl., Multi-platform, script-based user interface, CompSysTech, 2004.

[13] Giarratano, Joseph C., Ph.D. CLIPS User's guide Version 6.20, March 31st, 2002.

[14] Credit risk measurement and management according to the Basel II- http://www.eurorisksystems.com/download/CreditRiskBasel2_Eurorisk_EN.zip.

[15] Operational Risk Implementation based on Risk Framework- http://www.eurorisksystems.com/download/Presentation_OR_EuroRisk_EN.zip.

[16] Asset Liability Management in Risk Framework- http://www.eurorisksystems.com/download/Presentation_ALM_EuroRisk_EN.zip.

[17] Grady Booch, Robert Maksimchuk, Michael Engle, Bobbi Young, Jim Conallen, Kelli Houston, Object-oriented analysis and design with applications, third edition, Addison-Wesley Professional, 2007.

[18] 9th Intl. Protégé Conference - July 23-26, 2006 - Stanford, California- http://protege.stanford.edu/conference/2006/

[19] 10th Intl. Protégé Conference - July 15-18, 2007 - Budapest, Hungary- http://protege.stanford.edu/conference/2007/

[20] 11th Intl. Protégé Conference - June 23-26, 2009 - Amsterdam, Netherlands- http://protege.stanford.edu/conference/2009/

ABOUT THE AUTHORS

Dr. Anatoliy Antonov, Eurorisk Systems Ltd., 31, General Kiselov Str., 9002 Varna, Bulgaria, E-mail: antonov at eurorisksystems dot com

Samuil Nikolov, Eurorisk Systems Ltd., 31, General Kiselov Str., 9002 Varna, Bulgaria