Knowledge.NET ontology-based knowledge management toolkit for Microsoft.NET

Vladimir Safonov
St. Petersburg university
28 Universitetsky prospect
Petrodvorets
St. Petersburg 198504 Russia
v_o_safonov@mail.ru

Anton Novikov
St. Petersburg university
28 Universitetsky prospect
Petrodvorets
St. Petersburg 198504 Russia
antonnovik@gmail.com

Alexey Smolyakov
St. Petersburg university
28 Universitetsky prospect
Petrodvorets
St. Petersburg 198504 Russia
smlkvalex@mail.ru

Dmitry Cherepanov
St. Petersburg university
28 Universitetsky prospect
Petrodvorets
St. Petersburg 198504 Russia
hail@pochtamt.ru

Maxim Sigalin
St. Petersburg university
28 Universitetsky prospect
Petrodvorets
St. Petersburg 198504 Russia
max_ilya@mail.ru

ABSTRACT

The integration of knowledge engineering to software engineering is one of the most promising software development methodologies. It enables the combination of “traditional” programming features with knowledge engineering constructs in one intelligent solution, which is desirable in modern software development.

Most of knowledge engineering tools are isolated, limited and based on very specific languages such as clones of LISP[Car60a] or PROLOG[Der96a]. As such, they are prevented from integrating them with more conventional languages and application packages.

In contrast, our Knowledge.NET knowledge management toolkit for Microsoft.NET is based on a hybrid knowledge representation language, an extension of C# with knowledge engineering features – ontologies and rule sets.

The system can be used to develop knowledge libraries (bases) and intelligent solutions for a variety of problem domains. The knowledge can be developed using the Knowledge.NET system knowledge editor, or can be extracted from the Internet or any text files.

The system also provides a knowledge converter from Knowledge.NET format to more commonly used KIF (Knowledge Interchange Format).

Keywords
Microsoft.NET, knowledge management, ontology, rule set, C#, converter, knowledge extractor, knowledge editor, Knowledge Interchange Format, knowledge converter, Microsoft Visual Studio.NET 2005, add-in

1. INTRODUCTION

Knowledge engineering originated in 1950s as a part of Artificial Intelligence (AI). In general, it studies how to create expert systems that solve “creative” problems in certain problems domain [Saf91a]. In reality, many problems can’t be resolved by pure algorithms only. Problem solving can be also related to some hierarchical (conceptual), factual and heuristic knowledge.

Some of the popular methods of knowledge representation listed below:

- Productions – represents knowledge as a set of rules: IF condition THEN action. Rules are convenient for representing heuristic knowledge. Nowadays, PROLOG is the most common language to build rule-based expert systems;
- Frames – hierarchical knowledge structures. A frame has a set of slots and can be inherited from other frames. In turn, each of the slots has a value which

can be a value of simple type as well as a reference to another frame. The concept of frame was introduced by Marvin Minsky in 1970s [Min74a].

- Ontologies. The term *ontology* was borrowed from philosophy in early 1990s. The most known definition of ontology was formulated by Tom Gruber: "An ontology is an explicit specification of a conceptualization". More specifically, we define the ontology as a kind of specification of a problem domain in terms of its concepts and their relationships that allows to describe and to share hierarchical and factual knowledge in very efficient way. Knowledge Systems Laboratory of Stanford University supports visualization tool Protégé [Tut04a], which allows to manage ontology-based and frame-based knowledge.

Usually, knowledge engineering tool supports only one type of knowledge and has a very specific and limited semantic of base language, e.g. LISP, PROLOG. In our opinion, it increases complexity of usage of these tools. In contrast to many other tools, Knowledge.NET supports hybrid knowledge (productions, ontologies). More of that, as a base language, we use C# [Spe05a] extended by additional keywords and constructs for knowledge representation.

In section 2 we briefly describe principles of the Knowledge.NET system. In section 3 we present our approach to convert knowledge into KIF format. Section 4 briefly describes knowledge extractor subsytem. The summary section concludes and outlines the perspectives for the future.

2. PRINCIPLES OF KNOWLEDGE.NET

Seamless Integration to Visual Studio

Like Aspect.NET, the Knowledge.NET system is implemented as an add-in to Visual Studio.NET 2005. It actually means that, on installing the add-in, Knowledge.NET GUI becomes part of Visual Studio GUI. Due to that, it is possible to use all of VS.NET's wide spectrum tools and features for application design, implementation, debugging, profiling, etc., when developing an intelligent solution in Knowledge.NET. The add-in contains the following components:

- Knowledge Editor and Coloring;
- Coloring and IntelliSense for Knowledge.NET language;
- Add new type of projects: Knowledge.NET

Hybrid Knowledge Representation Language

The Knowledge.NET language is an extension of C# by constructs for representing hybrid, ontology-based knowledge. Semantics of ontology in Knowledge.NET is similar to that of the OWL knowledge representation language [Ove04a] developed by the W3C consortium.

A Knowledge.NET source code is at first converted into the ordinary C# code, and then compiled by the .NET C# compiler into an assembly that can be used as any other .NET application.

2.1.1 Program structure

A Knowledge.NET application source code consists of the following parts

- C# source code;
- Knowledge .NET – specific source code (concepts, properties, instances, rule sets)

The C# source code part is separated from Knowledge .NET source code by the "#ontology" keyword.

The user can use from her C# code concepts and their properties using standard way of addressing: [concept_name].[property_name]

Query language

The Knowledge.NET query language is one of the ways of accessing its knowledge base. It allows to select from ontology instances satisfying a given criterion (query).

As an example, consider a query on the ontology "Vehicles": "All the vehicles whose maximal speed is not more than 100 kilometers per hour":

Individuals of concepts Vehicle where HasMaximumSpeed <= 100

The result of the above request will be a set of instances of the Vehicle concept whose HasMaximalSpeed property's value is not greater than 100 km/hour.

The query language is supported by the QueryEngine class defined in the Knowledge.NET class library.

Rule sets

Besides conceptual knowledge representation, Knowledge.NET also allows to define heuristic knowledge using rule sets. Thus, the above mentioned hybrid knowledge framework is supported. The format of rule set is very close to that used in the KEE system. As a context of a rule set, an ontology is used which contains that rule set. To use rules, Knowledge.NET contains a classical style implementation of forward and backward chaining. If necessary, one can use her own inference engine by implementing the IProductionSystem interface.

Knowledge Editor

Besides the Knowledge.NET language, we also developed a Knowledge Editor (see fig. 1). The editor can be used to browse, update and enter knowledge. So, the knowledge engineer can work either in interactive mode or by creating knowledge directly in Knowledge.NET source. The knowledge editor also allows to navigate from graphical knowledge representation to textual knowledge representation and to call the converter from Knowledge.NET into C#.

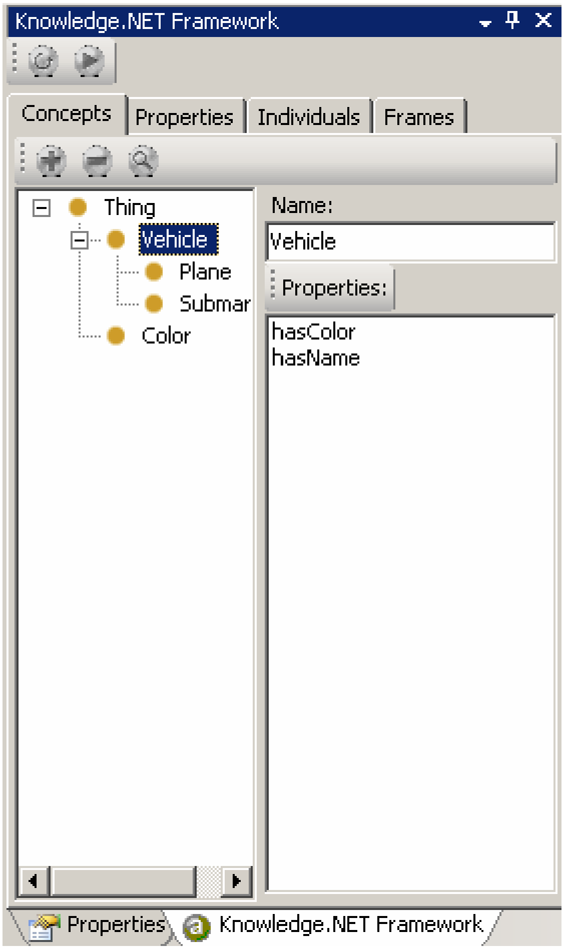

fig. 1

3. SUPPORTING FOR THE KIF FORMAT

Knowledge.NET also contains a converter of knowledge from Knowledge.NET to KIF (Knowledge Interchange Format) [Spe98a], to make the knowledge created in Knowledge.NET available for experts working in more traditional KIF format.

The conversion process consists of two stages. At the first stage the input document in Knowledge.NET is parsed and its internal representation is generated, in the format of the well known Ontolingua [Man97a] knowledge representation language. Actually at this stage the conversion process can be finished: the user can now use the output document in Ontolingua format. This approach is convenient because the user may convert this document into any of the knowledge representation languages supported by Ontolingua (in particular, to use the Ontolingua's KIF compiler).

At the second stage, the knowledge is converted into KIF format.

Overview of Ontolingua.

Ontolingua is a knowledge engineering environment containing a set of functions to work with ontologies (browsing, creating, updating and using). The Ontolingua language is a superset of KIF and contains constructs to represent frames and ontologies. The system also has a number of translators into other knowledge representation languages – Loop, Epikit, Express, Generic-Frame, Algernon, IDL and into KIF [Gru93a].

The Conversion Process

At the first stage, frames and concepts are converted into structures corresponding to Ontolingua's ***define-class*** and ***define-instance*** (for instances), properties – to ***define-relation***.

At the second stage, the following happens:

1. All the used relations are determined. Their representation in terms of KIF is written to the output stream.

2. define-relation and define-class are transformed into the DEFRELATION construct of the KIF language the following way:

:IFF-DEF replaced by ":="

:DEF replaced bt ":=>"

:CONSTRAINTS replaced by "=>"

:EQUIVALENT replaced by "<=>"

:AXIOM-DEF replaced by :AXIOM

3. For all primitive numbers used in the input Knowledge.NET source, the appropriate relations are generated (using the *defrelation* construct). The

following two KIF built-in numeric relations are used:

`integer` - expression`(integer t)` denotes that the *t* object is an integer;

`real` and `(real t)` – the same for real numberss.

For example, the byte type can be converted as follows:

```
(defrelation byte (?x) :=
  (and (natural ?x) (<= ?x 255)))
(defrelation natural (?x) :=
  (and (integer ?x) (>= ?x 0)))
```

4. DATA MINING

Knowledge.NET contains a knowledge extractor. This subsystem can be used to generate "raw" ontology from an array of documents or from some repository (e.g., from the Internet).

The scheme of functioning of the knowledge extractor is as follows:

- Morphological analysis of the input text and constructing a set of entities;
- Semantic analysis of the set of entities and constructing the knowledge graph;
- Analysis of the knowledge graph.

Morphological Analysis

At this stage, each word is converted to its *normal form* using MRD dictionaries and XML dictionaries.

Semantic Analysis

At this stage, a knowledge graph is constructed, with the nodes representing entities and edges representing relations. Analysis of entities is performed by customizable templates written in a specialized macro definition language.

Analysis of the Knowledge Graph

At this concluding stage:

- Different properties can be unified into one class. For example, the properties "big" and "giant" can be united into one class
- The graph is cleaned up to delete obsolete relations. For example, if an ascendant class has some property, the same property can be omitted from its descendant classes.

5. SUMMARY

Currently the work on Knowledge.NET is at the initial stage – the first prototype is developed and will be used to solve a variety of practical tasks that require using both "traditional" programming and knowledge engineering.

In future, we plan to integrate Knowledge.NET with Aspect.NET [Saf05a], to enable using Knowledge.NET for aspect-oriented knowledge engineering.

6. REFERENCES

1. [Gru93a] Gruber, T.R. A Translation Approach to Portable Ontology Specifications, 1993
2. [Man97a] Ontolingua Reference Manual at http://www-ksl.stanford.edu/htw/dme/thermal-kb-tour/ontolingua.html
3. [Ove04a] OWL Web Ontology Overview at http://www.w3.org/TR/owl-features/
4. [Saf91a] Safonov, V.O. TIP technology and its application in developing compilers and expert systems for high-performance computing systems. Doctoral dissertation, Leningrad, 1991
5. [Saf05a] Safonov, V.O. Aspect.NET: Aspect-Oriented Programming for Microsoft.NET in Practice. NET Developer's Journal, July 2005
6. [Spe98a] KIF language specification at http://logic.stanford.edu/kif/dpans.html
7. [Spe05a] C# Language Specification at http://msdn.microsoft.com/vcsharp/programming/language/#Language%20Specifications
8. [Tut04a] Protégé OWL Tutorial at http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf
9. [Min74a] Minsky, M. A Framework for Representing Knowledge, MIT-AI Laboratory Memo 306, 1974
10. [Car60a]McCarthy, J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I, April 1960
11. [Der96a]Deransart, P., Ed-Dbali, A., and Cervoni, L. Prolog: The Standard, 1996