Design Method for Knowledge Base Systems in Education Using COKB-ONT

Nhon Do,Tuyen Trong Tran, Phan Hoai Truong

***Abstract*—** Nowadays e-Learning is more popular, in Vietnam especially. In e-learning, materials for studying are very important. It is necessary to design the knowledge base systems and expert systems which support for searching, querying, solving of problems. The ontology, which was called Computational Object Knowledge Base Ontology (COB-ONT), is a useful tool for designing knowledge base systems in practice. In this paper, a design method for knowledge base systems in education using COKB-ONT will be presented. We also present the design of a knowledge base system that supports studying knowledge and solving problems in higher mathematics**.**

***Keywords*—**artificial intelligence, knowledge base systems, ontology, educational software.

I. INTRODUCTION

KNOWLEDGE representation has a very important role in designing knowledge base systems (KBS) and expert systems, especially those for education. There are many various models and methods for knowledge representation which have already been suggested and applied in many fields of science. Ontology is a new method which gives us a modern approach for designing knowledge components of KBS. However, practical applications of intelligent systems expect more powerful and useful models for knowledge representation. In this paper, we will use the ontology, which is called *Computational Object Knowledge Base Ontology* (COKB-ONT) [16], to produce an application in education and training. The COKB-ONT was used to produce many applications in education and training such as a program for studying and solving problems in plane geometry presented in [6], a system that supports studying knowledge and solving of analytic geometry problems presented in [7], and a knowledge base system in linear algebra, etc. These programs must have suitable knowledge base. They not only give human readable solutions but also present solutions as the way teachers and students usually write them. The COKB-ONT includes models, specification language and deductive methods. It is the new and modern method which is convenient for studying of users and for using by inference engine. Besides, problems are also modeled easily so that we can design algorithms for solving problems automatically and propose a simple language for specifying them. COKB-ONT is a natural and reasonable tool for designing knowledge bases. Nowadays, the COKB-ONT is often used to represent knowledge in different domain such as mathematics, physics, etc. In this paper, we will present a design method for knowledge base systems in education using COKB-ONT. It is also presented a case study, the design of a knowledge base system that supports studying knowledge and solving problems in higher mathematics.

II.COMPUTATIONAL OBJECT KNOWLEDGE BASE ONTOLOGY

There are many methods for knowledge representation [2], [4], [9] and [14]. These methods are interested and useful for many applications. However, they are not enough powerful and very difficult to use for constructing knowledge base systems in different domains of knowledge. The Computational Object Knowledge Base Ontology (COKB-ONT, [16]) and its models have been established from Object-Oriented approach to represent knowledge together with programming techniques for symbolic computation. There have been many results and tools for Object-Oriented methods, and some principles as well as techniques were presented in [15]. This way also gives us a method to model problems and to design algorithms. The models are very useful for constructing the components and the whole knowledge base of knowledge base systems in education.

A. Components of the COKB model

The model of computational object knowledge base (COKB model) consists of 6 components: **(C, H, R, Ops, Funcs, Rules)**. The meanings of the components are as follows:

- **C** is a set of concepts of computational objects (C-Object).
- **H** is a set of hierarchy relation on the concepts.
- **R** is a set of relations on the concepts.
- **Ops** is a set of operators.
- **Funcs** is a set of functions.
- **Rules** is a set of rules.

Each concept in C is a class of C-objects. The structure C-Objects can be modeled by **(*Attrs, F, Facts, Rules*)**. *Attrs* is a set of attributes, *F* is a set of equations called computation relations, *Facts* is a set of properties or events of objects, and *Rules* is a set of deductive rules on facts. An object also has basic behaviors for solving problems on its attributes. Objects are equipped abilities to solve problems such as:

1. Determines the closure of a set of attributes.
2. Executes deduction and gives answers for questions about problems of the form: determine some attributes from some other attributes.
3. Executes computations
4. Suggests completing the hypothesis if needed.

There are relations represent specializations between concepts in the set *C*; *H* represents these special relations on *C*. This relation is an ordered relation on the set C, and H can be considered as the Hasse diagram for that relation.

R is a set of other relations on *C*, and in case a relation r is a binary relation it may have properties such as reflexivity, symmetry, etc. In plane geometry and analytic geometry, there are many such relations: relation "belongs to" of a point and a line, relation "parallel" between two line segments, relation "perpendicular" between two line segments, the equality relation between triangles, etc.

The set *Ops* consists of operators on *C*. This component represents a part of knowledge about operations on the objects. Almost knowledge domains have a component consisting of operators.

The set *Funcs* consists of functions on C-Objects. Knowledge about functions is also a popular kind of knowledge in almost knowledge domains in practice, especially fields of natural sciences such as fields of mathematics, fields of physics.

The set Rules represents for deductive rules. The rules represent for statements, theorems, principles, formulas, and so forth. Almost rules can be written as the form "if <facts> then <facts>". In the structure of a deductive rule, <facts> is a set of facts with certain classification. Therefore, we use deductive rules in the COKB model. Facts must be classified so that the component *Rules* can be specified and processed in the inference engine of knowledge base system or intelligent systems.

Base on the COKB model, the knowledge base can be organized by the following components:

1. The dictionary of concepts about kinds of objects, attributes, operators, functions, relations and related concepts.
2. The table of descriptions for structures and features of objects. For example, we can request a triangle to compute and to give us its attributes.
3. The tables for representing hierarchical relations of concepts.
4. The tables for representing other relations of concepts.
5. The tables for representing knowledge about operators.
6. The tables for representing knowledge about functions.
7. The tables of descriptions for kinds of facts. For example, a relational fact consists of kind of the relation and the list of objects in the relation.
8. The tables of descriptions for rules. For example, a deductive rule consists of hypothesis part and conclusion part. Both of them are lists of facts.
9. The lists or sets of rules.
10. The lists of problem patterns.

B. Kinds of facts in COKB model

In the COKB model there are 11 kinds of facts accepted. These kinds of facts have been proposed from the researching on real requirements and problems in different domains of knowledge. The kinds of facts are as follows:

- **Fact of kind 1**: information about object kind.
- **Fact of kind 2**: a determination of an object or an attribute of an object.
- **Fact of kind 3**: a determination of an object or an attribute of an object by a value or a constant expression.
- **Fact of kind 4**: equality on objects or attributes of objects. This kind of facts is also popular, and there are many problems related to it on the knowledge base.
- **Fact of kind 5**: a dependence of an object on other objects by a general equation.
- **Fact of kind 6**: a relation on objects or attributes of the objects. In almost problems there are facts of kind 6 such as the parallel of two lines, a line is perpendicular to a plane, a point belongs to a line segment.
- **Fact of kind 7**: a determination of a function.
- **Fact of kind 8**: a determination of a function by a value or a constant expression.
- **Fact of kind 9**: equality between an object and a function.
- **Fact of kind 10**: equality between a function and another function.
- **Fact of kind 11**: a dependence of a function on other functions or other objects by an equation.

The above models and kinds of facts can be used to represent knowledge in practical applications. Unification algorithms of facts were designed and used in different applications.

C. Specification Language for COKB model

The language for the COKB model is constructed to represent the knowledge of the form COKB model. This language includes the following:

- A set of characters: letter, number, special letter.
- Vocabulary: keywords, names.
- Data types: basic types and structured types.
- Expressions and sentences.
- Statements.
- Syntax for specifying the components of COKB model.

III. DESIGN METHOD

In this section, we will present a process to construct a knowledge base system for solving some problems in higher mathematics. Besides, techniques in each phase will be presented also.

A. Structure of System

A system, which supports searching, querying and solving higher mathematics problem, has the structure of an expert system. We can design the system which consists of six components:

- The knowledge base.
- The inference engine.

- The explanation component.
- The working memory.
- The knowledge manager.
- The interface.

The figure 1 below shows the structure of the system.

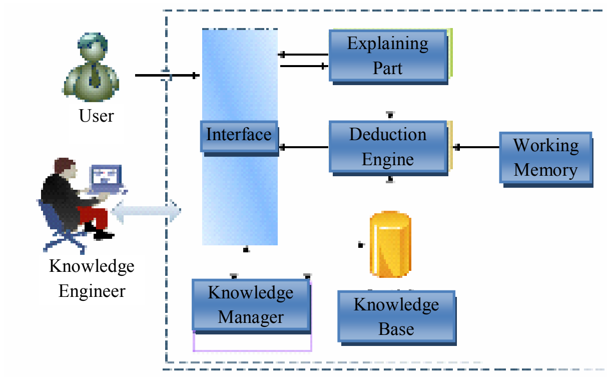

Fig. 1 Structure of system

The Knowledge Base contains the knowledge for solving some problems in a specific knowledge domain.

The Inference engine will use the knowledge stored in knowledge base to solve problems, to search or to answer for the query. It must identify problem and use suitable deductive strategies to find out right rules and facts for solving the problem.

The working memory stores the facts and rules in the process of searching and deduction.

The explanation component supports to explain the phases, concepts in the process of solving the problem.

The knowledge manager aims to support updating knowledge into knowledge base. It also supports to search the knowledge and test consistence of knowledge.

The interface component of the system is required to have a specification language for communication between the system and learners, between the system and teachers as well.

B. Design Technique

The process of analysis and design the components of the systems consists of the following stages.

Stage 1: Collecting real knowledge based on COKB-ONT model.

Stage 2: Classifying the knowledge in the Stage 1, to analyze requirements.

Stage 3: Establishing knowledge base organization for the system based on COKB-ONT model and its specification language. Knowledge base can be organized by structured text files. They include the files below.

- The file OBJECT_KINDS.txt stores names of concepts.
- The file HIERARCHY.txt stores information of the Hasse diagram representing for the component H of COKB model.
- The files RELATIONS.txt and RELATIONS_DEF.txt store the specification of relations (the component R of COKB model).
- The file OPERATORS.txt and the file OPERATORS_DEF.txt store the specification of operators (the component Ops of COKB model).
- The files FUNCTIONS.txt and FUNCTIONS_DEF.txt store the specification of functions (the component Funcs of COKB model).
- The file FACT_KINDS.txt stores the definition of kinds of facts.
- The file RULES.txt stores deductive rules.
- The file SOMEOBJECTS.txt stores certain objects.

Stage 4: Modeling of problems and designing algorithms. Problems are represented using a model that is called *networks of C-Objects*. It consists of three sets below.

$O = \{O_1, O_2, \ldots, O_n\}$,

$F = \{f_1, f_2, \ldots, f_m\}$,

$Goal = \{ g_1, g_2, \ldots, g_m \}$.

In the above model the set O consists of n C-objects, F is the set of facts given on the objects, and Goal consists of goals.

The design of deductive algorithms for solving problems and the design of interface of the system can be developed by three steps for modeling:

Step 1: Classify problems such as problems as frames, problems of a determination or a proof of a fact, problems of finding objects or facts, etc…

Step 2: Classify facts and representing them based on the kinds of facts of COKB model.

Step 3: Modeling kinds of problems from classifying in step 1 and 2. From models of each kind, we can construct a general model for problems, which are given to the system for solving them.

The basic technique for designing deductive algorithms is the unification of facts. Based on the kinds of facts and their structures, there will be criteria for unification proposed. Then it produces algorithms to check the unification of two facts.

The next important work is doing research on strategies for deduction to solve problems on computer. The most difficult thing is modeling for experience, sensible reaction and intuitional human to find the heuristics rules, which were able to imitate the human thinking for solving problems.

Stage 5: Creating a query language for the model. The language helps to design the communication between the system and users by words.

Stage 6: Designing the interface of software and programming the software. Our application has been implemented by using programming tools and computer algebra systems such as Visual Basic.NET or C#, SQL Server. They are very easy to use for students, to search, query and solve automatically problems.

Stage 7: Testing, maintaining and developing the application. This stage is similar as in other computer systems.

IV. CASE STUDY: DESIGN PARTERN – A KNOWLEDGE BASE SYSTEM FOR QUERYING KNOWLEDGE OF HIGHER MATHEMATICS

The software can support for searching, querying and solving some problems in higher mathematics. The first function of the program is "Search for Knowledge". This function helps users to find out necessary knowledge quickly. It can search for concepts, definitions, properties, related theorems or formulas, and problem patterns. Users declare some information base on a simple language. The information can consist of objects, relations between objects or between attributes. The program can automatically give instructions that help them to know more about the problem. The second function of the program is “Query for Knowledge”. This function helps users to query knowledge based on the query language which is a very simple language.

Stage 1: Collecting real knowledge based on COKB-ONT model.

The knowledge of higher mathematics consists of the following components.

A. Concepts:

- Numeric representation: natural numbers, integers, real numbers, complex numbers, etc.
- Category of sets: collection, Venn diagram, set operatoions, etc.
- Mapping: map, injection, surjection, bijection, identity mapping, inverse mapping, etc.
- Sequence of numbers: sequence, bounded sequence, bounded below sequence, bounded above sequence, Cauchy’s sequence, monotone sequence, Fibonacci sequence, pseudo random number sequence, strictly decreasing sequence, convergent sequence of function, convergence monotone sequence, double sequence, null sequence, decreasing sequence, random number sequence, strictly increasing sequence, limit of a sequence, etc.
- Real function with one variable: function, domain of a function, the range of a function, arithmetic function, even function, odd function, doubly periodic function, monotone function, polynomial, rational function, composite function, inverse function, reversible function, complex function, power function, exponential, Logarithmic function, trigonometric functions, inverse trigonometric functions, limit of a function, one-sides limit, limit of composite function, limit of monotone function, infinitesimal, infinitely great, continuity of function, continuous function at one point, one-sided continuous function, continuous function in a range, piecewise continuous function, point of discontinuity of function, function chart, maximum of a function, minimum of a function, concave of function, convexity of function, bending point of function chart, approximations of function chart, tangent line of function chart, etc.
- Derivation: derivation at a point, one-sided derivation at one point, derivation in a range, derivation of inverse function, high derivative, etc.
- Differential: differential invariant, differential at a point, differential on a range, primary differential, advance differential, etc.
- Initial function: complete primitive, primitive function calculus, etc.
- Integral: integral formula, integral domain, indeterminate integral, determinate integral, generalized integral with infinite bound (type 1 generalized integral), generalized integral of unbound function (type 2 generalized integral), integrate by parts, etc.
- Theory of series: series, convergent series, positive series, negative series, alternate series, random series, series of functions, domain of convergence of series of functions, convergence of series of functions, exponential series, radius of convergence of exponential series, Taylor's series of a function f(x) at adjacent point x_0., trigonometric series, Fourier’s series, Fibonacci series, etc.

B. **List of properties:** properties of series of convergence, bounded, order, etc.

C. **List of formulas:** Natural limit of series, Formulas for power function, Formulas for Logarithmic function, Taylor formula, MacLaurin’s theorem, L’Hospital rule, Derivation table, Cauchy-Schwarz’s inequality, Newton-Leibnitz formula, etc.

D. **List of computing method:** Function exploration, computing method of undetermined integral: analysis method, variable method, partial method, computing method of determined integral: analysis method, variable method, partial method finding radius convergence of power series, etc.

E. **List of theorems:** Infinitesimal and infinitely great theorem, Continuity theorem of function: Weierstrass1 theorem, Weierstrass2 theorem, Theorem of derivation, Mean theorem: Fermat, Rolle, Lagrange, Cauchy, L’Hospital 1 and L’Hospital 2 Theorems for function exploration, Theorem for initial function and integral, Series theorem, etc.

F. **Some applications of simply analytic function**: geometric meaning of derivation, application of determinate integral, etc.

Stage 2: Classifying the knowledge in the Stage 1, to analyze requirements.

- Basic concept: numeric representation, collection.
- Concepts are constructed from other concepts. For example: Map: collection; Function: map; Limit of series: series; Limit of function: function, limit of series; Chart of function: function, point; Derivation: function, limit of function; derivatives of higher order: derivation; Differential: function, derivation; Initial function: function, derivation; Integral: function, initial function; Series: array of number; Positive series: series; Alternative series: series; Series of functions: series, functions; Power series: series of functions; etc.

- Concepts are inherited from other concepts. For example: Injection: map; Surjection: map; bijection: map; Identity mapping: map; Function: map; etc.
- Attribute of concept. For example: Collection: number of element; Function: domain of determinacy, range of values; Chart of function: approximation, bend point; Power series: interval of convergence, radius of convergence; etc.
- Property of concept. For example: Collection: infinity, finite, empty; Series: ascending, descending, convergence, divergence; Function: ascending, descending, even, odd, convex, concave; etc.
- Operations on concepts. For example: arithmetic operations, collection operations, etc.
- Computing for functions. For example: limit of series, limit of function, inverse function, continuity of function, derivation, differential, initial function, integral, etc.

Stage 3: Establishing knowledge base organization for the system based on COKB-ONT model and specification language. Knowledge base can be organized by structured text files. They include the files below.

- The file OBJECT_KINDS.txt stores names of concepts.

```
begin_Objects
  <object name >
  <object name>
      ...
end_Objects
```

Example:

```
begin_Objects
      COLLECTION
      MAP
      SERIES
      FUNCTION
      DERIVATION
      INITIAL_FUNCTION
      ...
end_Objects
```

- The file HIERARCHY.txt stores the Hasse diagram representing for the component H of COKB model.

```
begin_Hierarchy
      [<high-order object>, <low-grade object>]
      [<high-order object>, <low-grade object>]
      ...
end_Hierarchy
```

Example:

```
begin_Hierarchy
      INJECTION, MAP
      SURJECTION, MAP
      BIJECTION, MAP
      IDENTITY_MAPPING, MAP
      POWER_FUNCTION, FUNCTION
      EXPONENTIAL, FUNCTION
      LOGARIT_FUNCTION, FUNCTION
      POSITIVE_SERIES, SERIES
      FUNCTIONS_SERIES, SERIES
      ...
end_Hierarchy
```

- The files RELATIONS.txt and RELATIONS_DEF.txt store the specification of relations (the component R of COKB model).

```
begin_Relations
[<relation name>,<object 1>,<object 2>],{<behavior>,< behavior >,...}
[<relation name>,<object 1>,<object 2>],{<behavior>,< behavior >,...}
   ...
end_Relations
```

Example:

```
begin_Relations
[CONSTRUCTION, COLLECTION, MAP],{}
[CONSTRUCTION, COLLECTION, FUNCTION],{}
[INHERITANCE, INJECTION, MAP],{}
[INHERITANCE, SURJECTION, MAP],{}
[INHERITANCE, BIJECTION, MAP],{}
[INHERITANCE, SERIES, ARRAY],{CONVERGENCE, DIVERGENCE}
[INHERITANCE, FUNCTION_POWER, FUNCTION],{ODD_EVEN}
   ...
end_Relations
```

- The files OPERATORS.txt and OPERATORS_DEF.txt store the specification of operators (the component Ops of COKB model).

```
begin_object: <Object name>
 begin_variables
     <attribute 1>
     <attribute 2>
     ...
 end_ variables
 begin_contains
 <path to file which stores object content>
 end_contains
end_object
```

Example:

```
COLLECTION.txt:
        begin_object: COLLECTION
        begin_variables:
          n: NUMBER_OF_ITEM
          A: ARRAY#LIST_OF_ITEM
        end_variables
        begin_contains
                 \Contains_Object\COLLECTION
        end_contains
        end_object
MAP.txt:
        begin_object: MAP
        begin_variables:
          X: COLLECTION_SOURCE
          Y: COLLECTION_DESTINATION
        end_variables
        begin_contains
          \Contains_Object\MAP
        end_contains
        end_object
....
```

- The files FUNCTIONS.txt and FUNCTIONS_DEF.txt store the specification of functions (the component Funcs of COKB model).

```
Begin_Functions
Begin_Function <name of function>([<list of arguments>])
    <argument>:<kind>
    Return <variable result>:<kind>
        Begin_description
    <path to the file which describes function>
    End_description
  End_Function
Begin_Function <name of function>([<list of arguments>])
    <argument>:<kind>
    Return <variable result>:<kind>
    Begin_description
    <path to the file which describes function>
    End_description
            End_Function
    ...
End_Functions
```

Example:

```
Begin_Functions
  Begin_Function INVERSE_FUNCTION(y)
  y: FUNCTION
```

```
    Return f: FUNCTION #f is the inverse function of y
    Begin_description
      \Contains_Object\INVERSE_FUNCTION
    End_description
    End_Function
    Begin_Function LIMIT_OF_ARRAY(d)
      d: ARRAY
      Return a: REAL
      Begin_description
      \Contains_Object\ LIMIT_OF_ARRAY
      End_description
    End_Function
    etc.
End_Functions
```

- The file FACT_KINDS.txt stores the definition of kinds of facts.

```
Begin_Methods
    Begin_Method < name of method>
      Begin_description
      <path to the file which describes method>
      End_description
    End_Method
    Begin_Method <name of method>
      Begin_description
      <path to the file which describes method>
      End_description
    End_Method
    ...
End_Methods
```

Example:

```
Begin_Methods
          Begin_Method INDETERMINATE_FORM_ELIMINATION
          Begin_description
            \Contains_Object\
          INDETERMINATE_FORM_ELIMINATION
          End_description
          End_Method
          Begin_Method FUNCTION_EXPLORATION_PROCESS
          Begin_description
            \Contains_Object\ FUNCTION_EXPLORATION_PROCESS
          End_description
          End_Method
          ...
End_Methods
```

- The file RULES.txt stores deductive rules.

```
Begin_Rules
    Begin_Rule <Rule name>:<kind of rule>
      Variables:
      <object>:<kind>
      <object>:<kind>
      ...
      Begin_description
      < path to the file which describes rules>
      End_description
      Goal:
        <result of using rule>
      End_Goal
    End_Rule
    Begin_Rule <Rule name>:<kind of rule>
      Variables:
      <object>:<kind>
      <object>:<kind>
      ...
      Begin_description
      < path to the file which describes rules>
      End_description
      Goal:
        <result of using rule>
      End_Goal
    End_Rule
    ...
End_Rules
```

- The file SOMEOBJECTS.txt stores certain objects.

Stage 4: Modeling for problems and designing algorithms. Problems are represented using a model that is called *networks of C-Objects*.

The design of deductive algorithms for solving problems and the design of interface of the system can be developed by three steps for modeling:

Step 1: Classify problems such as problems as frames, problems of a determination or a proof of a fact, problems of finding objects or facts, etc.

Step 2: Classify facts and representing them based on the kinds of facts of COKB model.

Step 3: Modeling kinds of problems from classifying in step 1 and 2. From models of each kind, we can construct a general model for problems, which are given to the system for solving them.

The basic technique for designing deductive algorithms is the unification of facts. Based on the kinds of facts and their structures, there will be criteria for unification proposed. Then it produces algorithms to check the unification of two facts.

The next important work is doing research on strategies for deduction to solve problems on computer. The most difficult thing is modeling for experience, sensible reaction and intuitional human to find the heuristics rules, which were able to imitate the human thinking for solving problems.

Stage 5: Creating a query language for the COKB-ONT. There are eight of simple query kinds and one of combine query kind:

- **Kind 1**: **?concepts|define** <*concept*>
  Example:
  ?concepts COLLECTION
  ?define FUNCTION
- **Kind 2**: **?attributes** <*concept*>
  Example: ?attributes COLLECTION
- **Kind 3**: **?properties** <*concept*>
  Example: ?properties CONVERGENCE_ARRAY
- **Kind 4**: **?formula** <*concept*>
  Example: ?formula LIMIT_OF_ARRAY
- **Kind 5**:
  *a)***?theorems** <*concept*>
  Example: ?theorems DERIVATE
  b) **?theorems** <*concept1, concept2,...*>
  Example: ?theorems FUNCTION, DERIVATE
  c) ?**content** <name of *theorem*>
  Example: ?content WEIERSTRASS_THEOREM
- **Kind 6**: **?methods** **<***concept***>**
  Example: ?methods LIMIT_OF_FUNCTION
- **Kind 7**: **? concepts_related** **<***concept*>
  Example: ? concept_object FUNCTION

➢ **Kind 8**: querying of many kind (kind 2 to kind 7)

```
q → kind 1..7
kind 8 →        q1  and  q2
             |  kind 81  and  kind 82
```

Example:

6

(? define LIMIT_OF_FUNCTION) and
(? properties LIMIT_OF_FUNCTION)

Annotation: All of concepts which is queried by using kinds of the query sentences, are in ***Objects.txt***.

Stage 6: Designing the interface of software and programming the software. Our application has been implemented by using programming tools and computer algebra systems such as Visual Basic.NET or C#, SQL Server. They are very easy to use for students, to search, query and solve automatically problems.

Stage 7: Testing, maintaining and developing the application. This stage is similar as in other computer systems.

V. CONCLUSION

The design method using COKB-ONT provided a natural way for representing knowledge for a class of knowledge base systems in education. We have designed the intelligent educational software for e-learning. Main functions of the software support for searching, querying and solving the problems in higher mathematics. Certainly, this design method can be used for applications in different domains of knowledge.

REFERENCES

[1] L. Stojanovic, J. Schneider, A. Maedche, S. Libischer, R. Suder, T. Lumpp, A. Abecker, G. Breiter, J. Dinger, The Role of Ontologies in Autonomic Computing Systems, TBM Systems Journal, Vol 43, No 3, 2004.

[2] Stuart Russell & Peter Norvig, *Artificial Intelligence – A modern approach* (second edition), Prentice Hall, 2003.

[3] John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*, Brooks/Cole, 2000.

[4] George F. Luger & William A Stubblefield, Artificial Intelligence, Addison Wesley Longman, Inc. 1998.

[5] Gruber, T. R., *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. International Journal Human-Computer Studies, 43(5-6):907-928, 1995.

[6] Do Van Nhon, A Program for studying and Solving of problems in Plane Geometry, Proceedings of International Conference on Artificial Intelligence 2000, Las Vegas, USA, 2000, pp. 1441-1447.

[7] Do Van Nhon, A system that supports studying knowledge and solving of analytic geometry problems, 16th World Computer Congress 2000, Proceedings of Conference on Education Uses of Information and Communication Technologies, Beijing, China, 2000, pp. 236-239.

[8] Asunción Gómez-Pérez & Mariano Férnandez-López & Oscar Corcho, *Ontological Engineering*. Springer-Verlag, 2004.

[9] Chitta Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.

[10] Guarino, N. *Formal Ontology, Conceptual Analysis and Knowledge Representation*, International Journal of Human-Computer Studies, 43(5-6):625–640, 1995.

[11] Wen-tsun Wu, *Mechanical Theorem Proving in Geometries*. Springer-Verlag, 1994.

[12] Chou, S.C. & Gao, X.S. & Zhang, J.Z. *Machine Proofs in Geometry*. Singapore: Utopia Press, 1994.

[13] Pfalzgraf, J. & Wang, D. *Automated Practical Reasoning*. New York: Springer-Verlag, 1995.

[14] Lakemeyer, G. & Nebel, B. *Foundations of Knowledge representation and Reasoning*. Berlin Heidelberg: Springer-Verlag, 1994.

[15] Berge, J.M. & Levia, O. & Rouillard, J. *Object-Oriented Modeling*. Netherlands: Kluwer Academic Publishers, 1996.

[16] Nhon Do, *An ontology for knowledge representation And Applications*. Waset, International Conference on Data, Information and Knowledge Management, Singapore, 2008.

Nhon Do is currently a senior lecturer in the faculty of Computer Science at the University of Information Technology, Ho Chi Minh City, Vietnam. He got his M.Sc. and Ph.D. in 1996 and 2002 respectively, from The University of Natural Sciences – National University of Ho Chi Minh City. His research interests include Artificial Intelligence, computer science, and their practical applications, especially intelligent systems and knowledge base systems.

Tuyen Trong Tran is currently a senior lecturer in the faculty of Information Technology at the Binh Duong University, Binh Duong Province, Vietnam. He got his M.Sc. in 2003 from The University of Natural Sciences – Viet Nam National University of Ho Chi Minh City. His research interests include Computer science, Management Information System.

Phan Hoai Truong is currently a senior lecturer in the faculty of Economics and Law at the Vietnam University of Ho chi minh city. He got his M.Sc. in 2002 from The University of Natural Sciences – Viet Nam National University of Ho Chi Minh City. His research interests include Computer science, Management Information System.