Ontology Engineering and Modelling for Learning Activity in a Multiagent System

Conference Paper · April 2014

DOI: 10.1109/SIMS.2014.35

3 authors:

Ken E Ehimwenma
Wenzhou-Kean University
16 PUBLICATIONS **57** CITATIONS
SEE PROFILE

Martin D. Beer
Sheffield Hallam University
105 PUBLICATIONS **921** CITATIONS
SEE PROFILE

Paul Crowther
Sheffield Hallam University
54 PUBLICATIONS **147** CITATIONS
SEE PROFILE

Ontology Engineering and Modelling for Learning Activity in a Multiagent System

Kennedy Ehimwenma, Martin Beer and Paul Crowther
Communication & Computing Research Centre
Sheffield Hallam University
Sheffield, United Kingdom
Email: Kennedy.K.Ehimwenma@student.shu.ac.uk, M.Beer@shu.ac.uk and P.Crowther@shu.ac.uk

***Abstract*—Prior and personalised learning is one area in cognitive learning that can be engineered on the platform of agent based intelligent systems. The requirement for inviting prior learning into a new learning context is the concept relationships between previous learning and the desired learning. In this paper this relationship has been established using ontology and mutliagent system in orchestrating a more personalised learning. This paper thus, present the use of Protege in the design of structured learning concepts in the domain of Computer Architecture. In this knowledge representation, attributes or properties are specified for classes, subclasses and individual members along with their constraints using Universal and Existential Restrictions. To the individuals, universal resource locator (URL) data values of the type string are assigned using the Data Property. And this process which has evolved in the development of a multiagent system for assessing prior learning before the take-off of new learning is being implemented with Jason AgentSpeak language.**

Keywords-ontology, computer architecture, triple, knowledge representation, learning, Pre-assessment, multiagent systems

I. INTRODUCTION

Ontology as a process of knowledge representation help us to visually represent knowledge domain, abstract concepts and the relationships that exist between concepts. With ontology tools, rules establishing relationships between objects or concepts are specified; and the properties shared are further controlled as constraints are being applied. Whether such constraints and property relationships are logically consistent is determined by the ontology Reasoner.

This paper is a preliminary work in the process of engineering an agent-based preassessment system. Following the class or concept relationships on the ontology in a top-down hierarchy approach, a learner would enter a concept as his desired learning target; the pre-assessment system perceives the open-ended input via percept in a multiagent system (MAS) environment. The learner is pre-assessed on the subclass concept (i.e. prerequisite) and depending on the outcome of pre-assessment, he will be tutored in his desired or pre-requisite concept on the system. We begin this paper by presenting the construction of Computer Architecture ontology with Protege 4.2.

In this presentation, a detailed concept of classes, subclasses and individual objects are constructed, typical of a school learning curriculum; in the order of simple to complex. The constructive analysis of the domain concepts from simple to complex underscores and espouse the common attributes or properties between concepts that can easily and ordinarily be undermined in a pedagogical paradigm; which are necessary relationships in the development of sequential learning and pre-assessment materials for agent-based intelligent systems. This paper continues with the meaning of ontology and agents in Section II. Section III presents the design phase of Computer Architecture (CA) ontology and its decision tree model. Section IV is Discussion and Section V conclusions and further work.

A. Related Work

Several ontology development approaches already exists in literature. There are for example, Commodity ontology [13], Marine Search and Rescue ontology [12]; and Research Activities Management ontology [2]. Specifically of interest to this work is the latter in which the ontology engineering process was extensively discussed using triples: *predicate[subject,object]*. But this has shortcomings in the specification of Existential and Universal constraints for property relationships and resources that are addressed in this paper. We also present a novel idea of decision tree modelling of ontology for learning activity in a multiagent based pre-assessment system.

II. ONTOLOGY

The essence of ontology is to specify true and valid relation or properties that exists between objects in a logical ideology. Gruber [4], [5] define ontology as a specification of conceptualization. Ontology specifies the classes of objects that exist, the relationships amongst those classes, the possible relationships amongst instances of the classes, and constraints over those instances. This specification or representation are typically classes (or sets), attributes (or properties), and relationships (or relations among class members) [6], [9] are possibly established using ontology editing tools. In formal concepts, an ontology is a 5-tuple O = < C; R; F; A; I > [8] where:

C: a finite set of named concepts organised in a taxonomy.

978-0-7695-5198-2/14 $31.00 © 2014 Crown Copyright
DOI 10.1109/SIMS.2014.35

R: a finite set of binary relations among concepts.
F: functions that relates concept and relations
A: a set of axioms that are valid in the conceptualisation.
I: a set of individuals belonging to a domain.

A. Agents

According to Wooldridge [11] an agent is a computer system that is situated in some environment. In that environment they exhibit some properties of autonomy, sociability, cooperation, etc., in order to meet their design objectives. They can observe and perceive the state of the environment that they are situated in, and in effect perform the actions assigned. Agents may not be stand-alone entity but a system consisting of a group of agents in the same environment otherwise known as multi-agent systems [3].

III. DESIGN PHASE

The Computer Architecture ontology is an educational ontology. Primarily, understanding the requirements of a domain is a crucial step in the ontology development process. Such understanding comes when the domain expert ponders over the type of ontology to build: General domain or domain specific ontology? Then build a list of vocabulary (concepts) and the relationships that holds between them. In that sense, this ontology is a domain specific ontology in which we have actively engaged the use of an ontology life cycle [1], [10].

A. Purpose

The name of this ontology is ComputerArchitecture (CA) developed for educational purposes. It is a structured hierarchy of concepts from simple to complex in an order that is analogous to a school learning curriculum, given using a top-down approach.

B. Scope

Computer Architecture is a field of computer science concerned with digital systems design such as logic design, registers, memory, control unit, instruction set, etc. To develop the CA ontology, the scope covers logic design which includes the operation of truth tables values represented by variables, Boolean logic, logic gate representation and the application of De Morgan's Theorem in digital circuits construction.

C. Data Capture (Classes and subclasses)

There are two disjoint super classes in the CA ontology, namely: ComputerArchitecture and Input_OutputVar classes. Below this are seven subclasses of the ComputerArchitecture super class. Using the Backus Naur Form (BNF), we have defined the classes as a set of elements of subclasses and individual members (Fig. 1) with their properties and constraints given in Table I and Table II.

Computer Architecture ::= {<ComputerArchitecture>,<Input_OutputVar>}

ComputerArchitecture ::= {<LogicGateCircuits_KB>,<LogicGates_KB>,<LogicGatesSymbols>,

<MorgansTheorem_KB>,<BooleanAlgebra_KB>,<TruthTable_KB>,<Operators>}

LogicGateCircuits_KB ::= {<MixedGatesCircuits_KB>,<SameGatesCircuits_KB>}

MixedGatesCircuits_KB ::= {MixedGatesCircuits_KB)

SameGatesCircuits_KB ::= {All_AND_Gate, All_NAND_Gate, All_NOR_Gate, All_OR_Gate, All_XNOR_Gate, All_XOR_Gate}

LogicGates_KB ::= { BasicGates_KB, DerivedGates_KB}

LogicGatesSymbols ::= {<BasicGatesSymbols>, < DerivedGatesSymbols>}

BasicGatesSymbols ::= {AND_Symbol, NOT_Symbol, OR_Symbol }

DerivedGatesSymbols ::= {NAND_Symbol, NOR_Symbol, XNOR_Symbol, XOR_Symbol}

MorgansTheorem_KB ::= {additionRule_KB, ProductRule_KB}

BooleanAlgebra_KB ::= {additionOperation_KB, ProductOperation_KB}

TruthTable_KB ::= {<MultiVariableInput>,<UnaryVariableInput>}

MultiVariableInput ::= {FourVariableInput, ThreeVariableInput, TwoVariableInput}

UnaryVariableInput ::= { UnaryVariableInput }

Operators ::= {<ArithmeticOperators>,<LogicalOperators>}

ArithmeticOperators ::= {AdditionOperator, ProductOperator}

LogicalOperators ::= {AND_Operator, NOT_ Operator, OR_Operator}

Input_OutputVar ::= {<UpperCaseVar>,<LowerCaseVar> }

UpperCaseVar ::= {A, B, C, … , X, Y, Z}

LowerCaseVar ::={a, b, c, … , x, y, z}

Fig.1: Set of classes and their members

TABLE I: CLASSES & SUBCLASSES,
RELATIONSHIPS AND CONSTRAINTS

Domain (Subject)	**Relationship (Predicate)**	**Constraint**	**Range (Object)**
ComputerArchitecture	has_KB has_Variable	Some some	ComputerArchitecture Input_OutputVar
LogicGateCircuits_KB	has_Prerequisite	only	LogicGates_KB
LogicGates_KB	has_Prerequisite	only	LogicGatesSymbols
LogicGatesSymbols	has_Prerequisite	only	DeMorgansTheorem
DeMorgansTheorem_KB	has_Prerequisite uses	only some	BooleanAlgebra_KB Not_Operator
BooleanAlgebra_KB	has_Prerequisite	only	TruthTable_KB
TruthTable_KB	has_Prerequisite	only	Operators
Operators	has_Prerequisite	only	Input_OutputVar
Input_OutputVar	IsUsedBy	only	ComputerArchitecture

D. Decision Tree

Having constructed the CA ontology, we developed a decision tree that models the ontology in order to analyse the process of pre-assessment in learning (Fig. 2) and the design process of the agent-based pre-assessment system. In this hierarchy of concepts, agents can pre-assess a learner on previous concept when a user enters a desired concept to

learn with the intention of identifying gaps in learning within the learner.

TABLE II: SUBCLASSES AND INDIVIDUAL (BOTTOM-LEVEL) MEMBERS AND INHERITED CONSTRAINTS

Subclasses & Individuals (Subject)	Relationship	Constraints	Object
MixedGatesCircuit_KB	Uses	some	BasicGatesSymbols
	uses	some	DerivedGatesSymbols
	uses	some	Operators
SameGateCircuits_KB			
- All_AND_Gate	uses	only	(AND_Operator and AND_Symbol)
- All_NAND_Gate	uses	only	(AND_Operator and NAND_Symbol and NOT_Operator)
- All_NOR_Gate	uses	only	(AdditionOperator and NOR_Symbol and NOT_Operator)
- All_OR_Gate	uses	only	(AdditionOperator and OR_Symbol)
	uses	only	(NOT_Symbol and XNOR_Symbol)
- All_XNOR_Gate	uses	only	XOR_Symbol
- All_XOR_Gate			
BasicGates_KB	Uses	some	BasicGatesSymbols
DerivedGates_KB	uses	some	DerivedGatesSymbols
BasicGatesSymbols			
- AND_Symbol	uses	only	AND_Symbol
- NOT_Symbol	uses	only	NOT_Symbol
- OR_Symbol	uses	only	OR_Symbol
DerivedGatesSymbols			
- NAND_Symbol	uses	only	(AND_Symbol and NOT_Symbol)
- NOR_Symbol	uses	only	(NOR_Symbol and NOT_Symbol)
- XNOR_Symbol	uses	only	NOR_Symbol
- XOR_Symbol	uses	only	OR_Symbol
AdditionRule_KB	Uses	some	OR_Operator
ProductRule_KB	uses	some	AND_Operator
AdditionOperation_KB	Uses	some	AdditionOperator
ProductOperation_KB	uses	some	ProductOperator
MultiVariableInput	Uses	some	Operators
- FourVariableInput	has_Prerequisite	only	Operators
	uses	some	Operators
- ThreeVariableInput	has_Prerequisite	only	Operators
	uses	some	Operators
- TwoVariableInput	has_Prerequisite	only	Operators
	uses	some	Operators
UnaryVariableInput	uses	only	NOT_Operator
AritmeticOperators	has_Prerequisite	only	Input_OutputVar
- AdditionOperator	has_Prerequisite	only	Input_OutputVar
- ProductOperator	has_Prerequisite	only	Input_OutputVar
LogicalOperators	has_Prerequisite	only	Input_OutputVar
- AND_Operator	has_Prerequisite	only	Input_OutputVar
- NOT_Operator	has_Prerequisite	only	Input_OutputVar
- OR_Operator	has_Prerequisite	only	Input_OutputVar
LowerCaseVar	isUsedBy	only	ComputerArchitecture
UpperCaseVar	isUsedBy	only	ComputerArchitecture

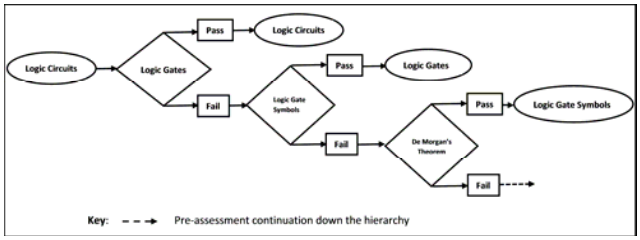

Fig. 2: Decision tree model of the CA ontology

IV. DISCUSSION

A. The semantics of CA Ontology

Properties are the entities that form relationships between concepts, and logically they depicts the meaning in a given relation. In Protégé ontology, there are three basic properties which are: Annotation property, Object property, and Data property. Properties have Domain and Range. So properties (i.e. predicate) link individuals from the Domain (the subject) to individuals from the Range (object) in Table I and Table II.

1) Annotation property: These are the properties meant to add comments to concepts, to describe their terms and relationships. For example, the *All_OR_Gate* comment description (Fig 3) that states:

All_OR_Gate circuit is constructed with the OR Gate Symbol and Addition Operator.

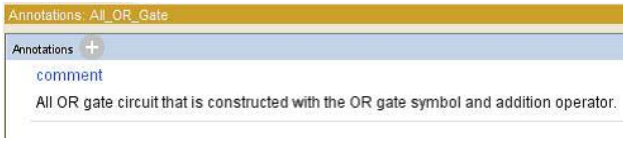

Fig. 3: Annotation property

2) Object property: Are properties used to establish relationships between classes (top-level concepts), subclasses (middle-level concepts) and individuals/instances (bottom-level concepts). In this work, we have defined our object properties as: *has_KB, hasPrerequisite, hasVariable*, and *uses*; as links between Domain and Range. And an *isUsedBy* inverse as a link between the Range and Domain, where *isUsedBy* is the inverse property of *uses*.
For example, the *AdditionRule_KB* concept (the subclass) of the *DeMorgansTheorem* class in Figure 1 can be stated in a triple i.e. <subject>-<predicate>-<object> as:

(AdditionRule_KB uses OR_Operator);

or in formal semantic or logic-based programming syntax as:

uses(AdditionRule_KB, OR_Operator)

where the Domain is *AdditionRule_KB* and *OR_Operator* is the Range. On the inverse, with the *isUsedBy* property, we have the statement:

isUsedBy(OR_Operator, AdditionRule_KB)

with the same parameters but the *OR_Operator* and *AdditionRule_KB* as Domain and Range, respectively. In a different perspective, a concept can be both Domain and Range in a given relation. For example,

has_KB(ComputerArchitecture, ComputerArchitecture)

is a valid relation since there are subclasses in the *ComputerArchitecture* class. These subclasses has knowledge base *ComputerArchitecture*, thus making the *ComputerArchitecture* class a Domain as well as a Range, where all its subclasses uses the *Input_OutputVar* class. To this effect, the *hasVariable* as well as the *has_KB* properties becomes inherited properties from the *ComputerArchitecture* super class. On the *hasVariable* property, at least (in digital hardware/circuit design) one

variable must be used to specify either input or output data. Hence, the relationship of all individual members with this properties in Table I. Still on Table I, it will be noticed that, in order to model knowledge for learning, concepts are dependent on each other. That is, a prerequisite (lower_level) concept must be understood before the immediate higher-level concept can be learnt. This relationship of knowledge synthesis is given as follows in triples in the form, *predicate[subject, object]:*

hasPrerequisite(LogicGateCircuit_KB, LogicGates_KB)
hasPrerequisite(LogicGates_KB, LogicGatesSymbols)
hasPrerequisite(LogicGatesSymbols, DeMorgansTheorem_KB)
hasPrerequisite(DeMorgansTheorem_KB, BooleanAlgebra_KB)
hasPrerequisite(BooleanAlgebra_KB, TruthTable_KB)
hasPrerequisite(TruthTable_KB, Operators)
hasPrerequisite(Operators, Operators)

3) Data property: These are bottom-level properties that describes the relationships between the class instances and their data values—that could be XML Schema Datatype or an RDF literal. In the CA ontology, we have used one data property: *hasContent* with an inverse of *isContentOf*, created for data values of the type String. This property holds the value of the respective URL addresses of the various webpages assigned that contains the learning-content materials for each individual concept. An example is that of the *AND_Symbol* in Figure 4 specified as:

hasContent(AND Symbol, http://hyperphysics.phyastr. gsu.edu/hbase/electronic/and.html#c);

with inverse of:

isContentOf(http://hyperphysics.phyastr.

gsu.edu/hbase/electronic/and.html#c, AND Symbol).

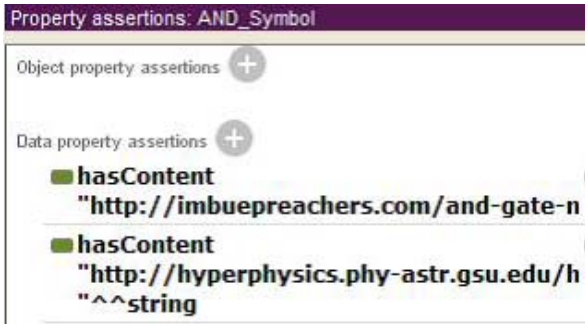

Fig. 4: Dataproperty specification

B. Property Constraints

These are further controls and restrictions to the relationship an object can participate in after properties are defined. That is why, they are specified alongside object properties (for example *uses*) that is already existing. Thus, at the design phase, the constraints were included, for class or class members to have existential (*some*) or universal (*only*) restriction.

1) Existential Restrictions: They describes classes of individuals that participate in at least one (minimum) relationship [7]. They are denoted by the term *some* (Figure 5) and specifies the satisfaction of a necessary condition by that class member in order to participate in that property or relation. For example, our *uses(AdditionRule_KB, OR_Operator)* has been extended to have the existential restriction (*some*) to produce the following restricted relation:

some(uses(AdditionRule_KB, OR_Operator));

which states, in the operation of the addition rule in *De Morgan's Theorem*, at least one OR operator is used.

2) Universal Restrictions: Are the *AllValueFrom* restrictions that is denoted by the *only* keyword. They are those restrictions that constrain the relationship of a given property to individuals that are members of a specific class or to individuals that would not also have a property relationship to individuals that are not members of the given class [7]. Consider the statement for the *All_OR_Gate* circuit under the *SameGateCircuits_KB* class:

only(uses(All_OR_Gate, (AdditionOperator and OR_Symbol)));

this fact reads that, in the construction of the *All_OR_Gate* circuit, only addition operator and the OR symbol are used (Figure 5).

Fig. 5: Object property, Existential and Universal constraints

C. Agent Based Pre-assessment System

Using Jason AgentSpeak language, we created five agents in a multiagent system called Agent Based Preassessment & Tutoring System (details outside the scope of this paper), and demonstrated a process of pre-assessment of learning concepts that mirrors the CA ontology as analysed in the decision tree of Figure 2. The agInterface agent that is situated in the text-area graphical user interface (GUI) gets percept (input signals) from the environment, and communicates to other agents. For example, the Boolean Algebra concept was entered as a learning target. The system tested the user (the designers) on the pre-requisite concept to the desired (Boolean Algebra) concept entered to learn. If the user response was right, a pass would be given, and the user would learn Boolean Algebra. But in this case, since the response given was wrong, the user got a fail. And according to the CA ontology hierarchy, the prerequisite to Boolean Algebra is the Truth-Table (the least concept because Operators are actually symbols), so the user was recommended to begin learning from the Truth-Table (Fig. 6). Otherwise, he would have been pre-assessed on the next immediate-lower concept.

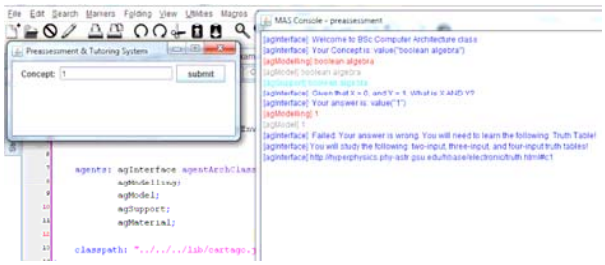

Fig. 6: Activities of User engagement with the Agent based Preassessment System.

V. CONCLUSIONS AND FUTURE WORK

This paper has presented the construction of a domain specific *ComputerArchitecture* (CA) ontology, its properties and constraints between concepts using Protégé. Then the modelling of the CA classes in a decision tree as inter-dependent entities especially in human learning. The CA ontology has emphasised a system of relationships in knowledge modelling that underlined prerequisite knowledge as building blocks for higher-level learning. That is, before the learning of a higher concept the immediate lower concept must be understood. To that effect, we used an Agent Based Pre-assessment & Tutoring System to demonstrate the pre-assessment of prior knowledge in order to identify any gaps in learning before the start of a new learning. The next stage of this work will be the use a multiagent system to demonstrate a system of semantic relationships of the CA ontology.

REFERENCES

[1] Jonathan DiLeo, Timothy Jacobs, and Scott DeLoach, "Integrating ontologies into multiagent systems engineering," Technical report, DTIC Document, 2006.

[2] Amjad Farooq, Abad Shah, and Khadam Hussain Asif, "Designofontology in semantic web engineering process," in High Capacity Optical Networks and Enabling Technologies, 2007, (HONET 2007), International Symposium on, pages 1–6. IEEE, 2007.

[3] Anatoly Gladun, Julia Rogushina, Rodrigo Mart´ınez-B´ejar, Jesualdo Tom´as Fern´andez-Breis, et al., "An application of intelligent techniques and semantic web technologies in e-learning environments," Expert Systems with Applications, 36(2):1922–1931, 2009.

[4] Thomas R Gruber, "A translation approach to portable ontology specifications," Knowledge acquisition, 5(2):199–220, 1993.

[5] Thomas R Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," International journal of human-computer studies, 43(5):907–928, 1995.

[6] Tom Gruber, "What is an ontology," 1993.

[7] Matthew Horridge et al., "A practical guide to building owl ontologies using prot´eg´e 4 and co-ode tools," edition1. 2, The University Of Manchester, 2009.

[8] Alexander Maedche and Steffen Staab, "Ontology learning for the semantic web," IEEE Intelligent systems, 16(2):72–79, 2001.

[9] Jan Morbach, Andreas Wiesner, and Wolfgang Marquardt, "Ontocapea (re) usable ontology for computer-aided process engineering," Computers & Chemical Engineering, 33(10):1546–1556, 2009.

[10] Mike Uschold and Michael Gruninger, "Ontologies: Principles, methods and applications," The knowledge engineering review, 11(02):93–136, 1996.

[11] Michael Wooldridge, "An introduction to multiagent systems," John Wiley & Sons, 2009.

[12] Weihong Yu and Ruixin Li, "Maritime search and rescue ontology construction based on prot´eg´e," in Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference, pp. 1–3. IEEE, 2009.

[13] Huiqun Zhao, Shikan Zhang, and Junbao Zhao, "Research of using prot´eg´e to build ontology," in ACIS-ICIS, pp. 697–700, 2012.