No More "Keyword Search" or FAQ: Innovative Ontology and Agent Based Dynamic User Interface

Nelson K. Y. Leung and Sim Kim Lau

Abstract—**The establishment of help desk is to provide technical support to users when they encounter technical problems related to hardware, software, application programs and network connections. However, due to resources problem, in particular the lack of help desk staff, users often have to wait for a considerably long time before their enquiries and problems are answered and solved. To relieve user's dissatisfaction, academic researchers and help desk practitioners has started to encourage users to exploit online resources. Instead of contacting help desk, user can access online knowledge resources, such as knowledge base, to look for information that is useful to resolve their existing difficulties. A significant number of knowledge bases are designed to support "keyword search" and FAQ as the front end user interface, but both designs have their disadvantages. This paper investigates the application of ontology and software agent technology to develop a dynamic user interface. This interface is designed to replace "keyword search" and FAQ as the front end user interface of the online knowledge base.**

Index Terms—**Dynamic User Interface, Ontology, Software Agent Technology.**

I. Introduction

Help desk also known as computer call centre, contact centre, assist centre or support centre is an access point to provide IT-related advice, information or troubleshooting action to user. In the past three decades, organizations have been investing heavily in IT and business information systems development to solve business problems, to gain competitive advantage and to sustain organizational improvement. However, the complexity of the business systems has created infinite number of technical and functional problems. This complexity also means that users are not able to work at optimal productivity when they come across technical problems related to the system. Organization may face potential loss in income, whether direct or indirect, immediate or in the future. The establishment of help desk is to provide technical support to users when they encounter technical problems related to hardware, software, application programs and network connections. Its responsibilities include first line incident support, day to day communication between Information Technology (IT) department and user, business systems support and service quality report generating [4,9].

Nelson K. Y. Leung is with the School of Information Systems, University of Wollongong, Wollongong, NSW, 2522, Australia (email: knl164@uow.edu.au).
Sim Kim Lau is with the School of Information Systems, University of Wollongong, Wollongong, NSW, 2522, Australia (email: simlau@uow.edu.au).

Due to continuous expansion of the "already complex" IT infrastructures, help desks are required to cover more and more software, hardware, network and other IT related areas. It is not unusual for a single help desk to cover hundreds of thousands of IT related products. On the other hand, downsizing and business process reengineering have led to the shrinkage of the size of help desk because its overall budget has been reduced. When help desk is expected to provide more service with less staff, the outcome is quite obvious: users often have to wait for a considerably long time before their enquiries and problems are answered and solved [6]. To relieve user's dissatisfaction, academic researchers and help desk practitioners has started to promote the concept of e-support. Broome and Streittwieser [3] describe all support actions that use Internet or web as the primary communication channel to be included in e-support. Instead of contacting help desk, users can access online knowledge resources, such as knowledge base, to look for information that is useful to resolve their existing difficulties.

A significant number of knowledge bases are designed to support "keywords search" as the front end user interface in which users can locate the most appropriate solutions by entering a few keywords that best describe the problems. However, users often do not know how to use the right jargons to explain the problems. Although they may successfully use their own words to depict the problems, the "keyword search" may return ten or even more solutions which will deepen users' frustration. The complexness of the user interface can drive away novice users or even users classified as medium-skilled. Another common user interface for knowledge base is Frequent Asked Question lists (FAQ). FAQ is always overlooked by users because its mechanism lacks the ability to support users in dynamic and flexible manners. This paper investigates the application of ontology and software agent technology to develop a dynamic user interface. This interface is designed to replace "keyword search" and FAQ as the front end user interface of the online knowledge base.

The rest of paper is organized as follows. Section 2 discusses the development of a dynamic user interface. This includes a discussion of ontology and software agent design. Illustration of dynamic user interface is presented in Section 3. Finally, conclusion is given in Section 4.

II. Development of Dynamic User Interface

The target users of the online knowledge base are users with low to medium technical skill. Therefore the design of the interface must be simple and user friendly. Subsequently, an easy to use dynamic user interface with interactive communication capability is proposed. The dynamic user interface allows users to present and identify the problems by choosing problem types and their symptoms from a series of drop boxes. In summary, the user interface provides two subsets of functionalities:

1. the functionality to browse help desk ontology.
2. the functionality to view solution from knowledge base based on results of ontology browsing.

Figure 1 shows the basic architecture of the dynamic user interface. There are four basic components within the architecture: user's browser, interface agent, ontology and knowledge base which stores solutions for users' problems. Here, a series of interrelated vocabularies which allows user to identify and describe their problems on the user interface, is stored and organized in a structural hierarchy within the ontology. Modern web technology is used as a means to deliver the interface through the Internet and can appear on the browser to facilitate the interaction with the user and deliver user request for resolution. On the other hand, software agent technology is used to facilitate user communication. Software agent is a computer program that behaves like human and is capable of autonomous actions in pursuit of specific goal [8,11]. To deliver the dynamic user interface, user simply clicks on the target Uniform Resource Locator (URL). Subsequently, the interface agent that possesses communication capability will deliver a dynamic user interface to the browser, based on the information stored in the ontology. The dynamic and interactive communication capabilities of the interface agent help users to identify and present their problems. Firstly the interface agent interacts with the user by asking the user to select a problem type on the user interface. Based on the input, the interface agent will generate the next category of possible problem scenarios from the ontology. This type of interaction will continue until the agent has gathered sufficient information. When the problem is completely described on the interface, the knowledge base will then deliver a related solution to the user. Since each set of problem descriptions is linked to a particular solution in the knowledge base, it guarantees the return of the most appropriate solution.

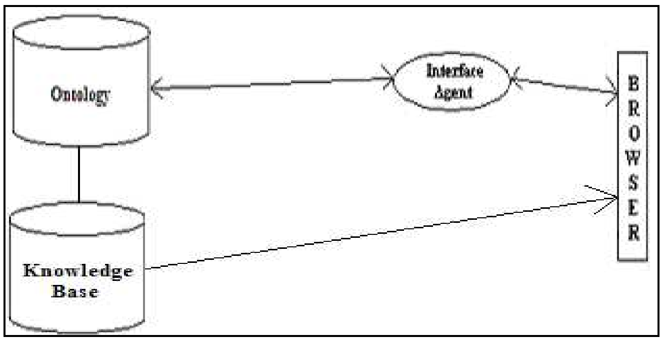


Figure 1 Basic Architecture of Dynamic User Interface

A. *Ontology Design*

Traditionally, the term "ontology" is defined as the study or the science of being. Gruber and Olsen [5] first apply ontology to artificial intelligence as the specifications of common conceptualizations among agents. In other words, agent is able to understand the semantic of other knowledge since knowledge is represented by the same vocabulary based on common conceptualization. The emergence of semantic web further magnifies the importance of ontology. Berners-Lee, Hendler and Lassila [2] recognize that the Hypertext Markup Language (HTML)-based web content is solely designed for human to read and computers have no way to understand and process the semantics. In the context of the web, ontology provides a shared understanding of a domain that contains a finite list of terms and the relationships [1]. In this way, an ontology enables computer programs and software agents to understand the semantics, thus making it possible for them to process the web content. Although different organizations may have their own ontologies, such differences can be overcome by mapping the particular terminology to a shared ontology or by defining direct mappings between ontologies [1].

In the dynamic user interface, a Web Ontology Language (OWL)-based ontology is developed to represent various categories of problem types and their symptoms. The problem types and symptoms are used to support the dynamic interface on which users can choose to describe and identify the problems. OWL builds on Resource Description Framework (RDF) and RDF Schema and adds more vocabulary for describing properties and classes, among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes [10]. The RDF uses Extensible Markup Language (XML) as interchange syntax to provide a lightweight ontology system to support the exchange of knowledge on the Web [1,7]. The ontology of this interface consists of two major categories. The first category describes the taxonomy of possible problem types, and the second depicts the taxonomy of symptoms in accordance with the problem types. Figure 2 depicts the problem types category and some of its subclasses. The problem types category has *Help_Desk_Enquiry* as its superclass. *Help_Desk_Enquiry* is then extended into four subclasses that include *IT_Administrative_Issue*, *Software_Problem*, *Hardware_Problem* and *Other_Problem*. These four subclasses are designed to represent the four main sources of problems. Further expansion of subclass and instance for each subclass is required until there is enough vocabulary to describe the problems.

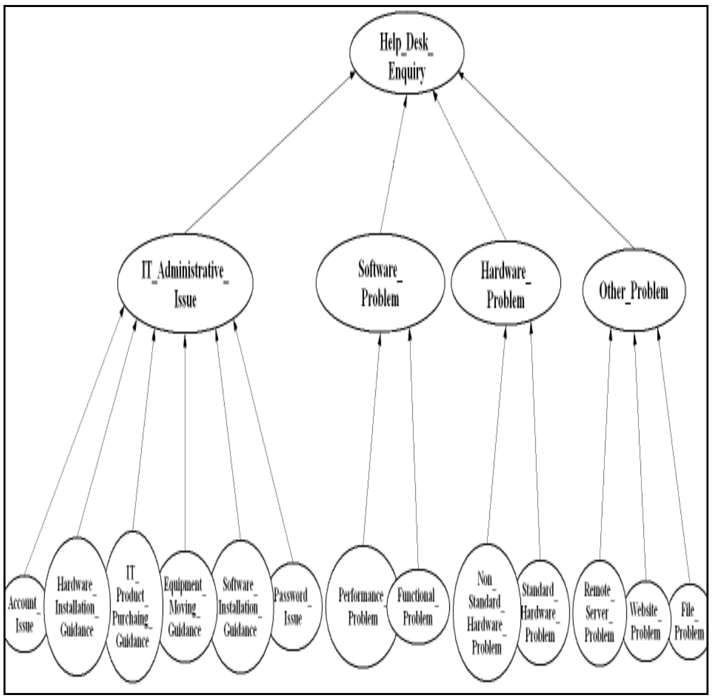


Figure 2 Problem Types Category and its Partial Subclasses

Figure 3 illustrates an example of the problem symptoms category and some of its subclasses. The problem symptoms class starts with *Problem_Symptoms* as its superclass. However, the expansion of this category is closely related to the problem types category. For example, *IT-Administrative_Issue_Symptom*, *Software_Problem_Symptom*, *Hardware_Problem_Symptom* and *Other_Problem_Symptom* are used to identify the problem symptoms of *IT_Administrative_Issue*, *Software_Problem*, *Hardware_Problem* and *Other_Problem* in the problem types category. The expansion of the problem symptoms category will continue until it is sufficient to identify all of the problem symptoms. Since problem types and problem symptoms are not standalone categories, every object in the problem types category are connected with an identical objects in the problem symptoms category by object properties. In OWL, object property is used to relate objects to other objects.

In Figure 4, object property *hasFileSymptom* and its inverse, *isFileSymptomOf*, is utilized to relate *File_Problem* with *File_Problem_Symptom*. This indicates that *File_Problem has_File_Symptom*, whereas *File_Problem_Symptom* is a symptom of *File_Problem*. Furthermore, the entire set of object properties and their inverses are organized in a hierarchy by using the concepts of property, subproperty and superproperty. For example, is*SymptomOf* has *isOtherProblemSymptomOf* and *isFileSymptomOf* as its subproperties. In other words, *isFileSymptomOf* has *isOtherProblemSymptomOf* and *isSymptomOf* as its superproperties. Figure 5 shows a partial hierarchy of the properties and their inverses.

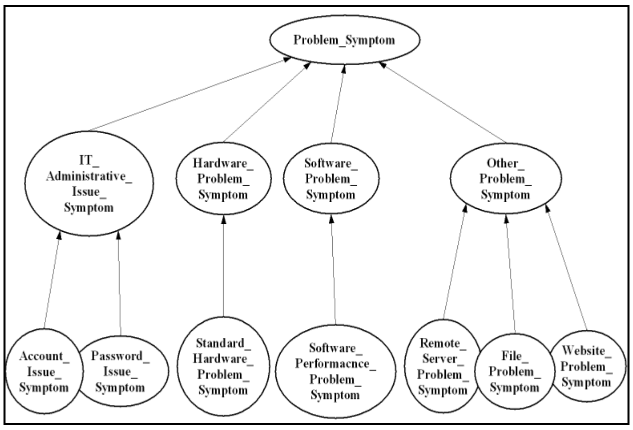


Figure 3 Problem Symptoms Category and its Partial Subclasses

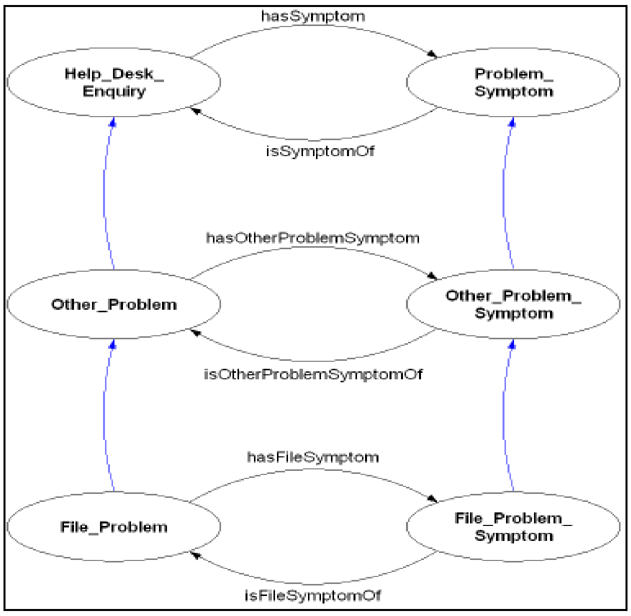
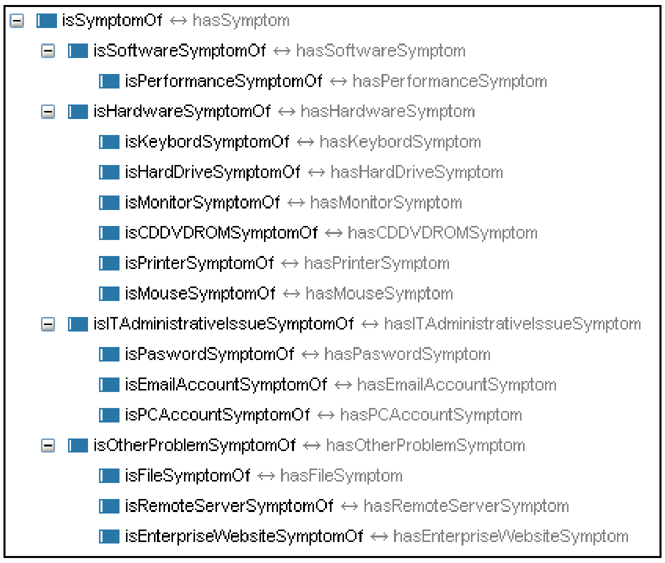


Figure 4 Semantic Relationships among Problem Types, Symptoms and Properties

To understand how the ontology could support the dynamic interface, let us consider one branch of problem types and its corresponding branch of problem symptoms (see Figure 4). *Help_Desk_Enquiry* is the superclass of *Other_Problem* and *File_Problem*. *Other_Problem* is a subclass of Help_*Desk_Enquiry* and has *File_Problem* as its subclass. *File_Problem* is a subclass of *Other_Problem* as well as *Help_Desk_Enquiry* and it does not have any subclass. In the property hierarchy, *hasSymptom* is the superproperty of *hasOtherProblemSymptom* and *hasFileSymptom*. In term of subproperty, *hasOtherProblemSymptom* is the subproperty of *hasSymptom* and *hasFileSymptom* as its subproperty. On the other hand, *hasFileSymptom* has no subproperty, but with *hasSymptom* and *hasOtherProblemSymptom* as its superproperty. Subsequently, *File_Problem* can have instances of *File_Problem_Symptom* as values because *hasFileSymptom* and its reverse relate these two subclasses together. In this case, the instances of *File_Problem_Symptom* are *File_Corrupted*, *File_Accidentally_Deleted*, *File_Accidentally_Modified* and *Missing_File*. Besides, the concepts of the subclass, superclass, superproperty and subproperty allow *File_Problem* to inherit *hasSymptom*, *hasOtherProblemSymptom* as its own properties. The same concept also applies to *File_Problem_Symptom* that inherits *isSymptomOf* and *isOtherProblemSymptomOf* as its own properties.



- isSymptomOf ↔ hasSymptom
  - isSoftwareSymptomOf ↔ hasSoftwareSymptom
    - isPerformanceSymptomOf ↔ hasPerformanceSymptom
  - isHardwareSymptomOf ↔ hasHardwareSymptom
    - isKeybordSymptomOf ↔ hasKeybordSymptom
    - isHardDriveSymptomOf ↔ hasHardDriveSymptom
    - isMonitorSymptomOf ↔ hasMonitorSymptom
    - isCDDVDROMSymptomOf ↔ hasCDDVDROMSymptom
    - isPrinterSymptomOf ↔ hasPrinterSymptom
    - isMouseSymptomOf ↔ hasMouseSymptom
  - isITAdministrativeIssueSymptomOf ↔ hasITAdministrativeIssueSymptom
    - isPaswordSymptomOf ↔ hasPaswordSymptom
    - isEmailAccountSymptomOf ↔ hasEmailAccountSymptom
    - isPCAccountSymptomOf ↔ hasPCAccountSymptom
  - isOtherProblemSymptomOf ↔ hasOtherProblemSymptom
    - isFileSymptomOf ↔ hasFileSymptom
    - isRemoteServerSymptomOf ↔ hasRemoteServerSymptom
    - isEnterpriseWebsiteSymptomOf ↔ hasEnterpriseWebsiteSymptom

Figure 5 Partial Hierarchy of Properties and their Inverses

B. *Software Agent Design*

The InterfaceSoftwareAgent is an agent that possesses communication capability and is in charge of providing vocabulary of problem types and symptoms on the user interface, based on the concept stored in the ontology. The vocabulary is to be used by users to describe the problems. When user clicks on the required link to access the interface, it will activate the InterfaceSoftwareAgent. Then, the InterfaceSoftwareAgent starts to retrieve and capture vocabulary in the ontology in accordance with the selections of the drop boxes selected. The online knowledge base will return an appropriate solution to user if all of the vocabularies related to a particular problem type and symptom have been retrieved from the ontology. The retrieval and reasoning capabilities are based on a set of rules:

1) *Continue to capture and display all direct subclasses in the drop box, based on user's selection that relates to their superclass in the problem types category.*
2) *If there is no related subclass, capture and display all related instances from the last selected class in the problem types category. The InterfaceSoftwareAgent will terminate and the online knowledge base will return an appropriate solution to users.*
3) *If there is no related subclass and instance from the last selected class in the problem types category, the InterfaceSoftwareAgent will examine all the object properties (includes all the inherited superproperties) that the last selected class in the problem types category possesses. This determines whether the direct connected class from other categories (categories other than the problem types) contains any instances.*
   a) *If there is an instance in one of the direct connected class, the InterfaceSoftwareAgent will capture and display all the instances in the drop box. The InterfaceSoftwareAgent will terminate and the online knowledge base will return an appropriate solution to users.*
   b) *If there is no instance in any of the direct connected classes, the InterfaceSoftwareAgent will terminate and the online knowledge base will return an appropriate solution to users.*
4) *If there is no related subclass, instance and object property from the last selected class in the problem types category, the InterfaceSoftwareAgent will terminate and the online knowledge base will return an appropriate solution to users.*

Let us consider Figure 6 and 7 as an example to demonstrate the rules of the InterfaceSoftwareAgent. Using rule 1, the InterfaceSoftwareAgent starts by capturing *Hardware_Problem*, *Software_Problem*, *IT_Administative_Issue* and *Other_Problem* based on the default superclass, *Help_Desk_Enquiry*, from the problem types category of the ontology. The four subclasses are displayed in the first drop box. User then decides to choose *Hardware_Problem* in the first drop box. Simultaneously, the interface-software agent captures *Non_Standard_Hardware_Problem* and *Standard_Hardware_Problem* from the problem types category of the ontology based on user's selection in the first drop box and display these two items in the second drop box (rule 1). Subsequently, the user decides to select *Non_Standard_Hardware_Problem* in the second drop box. The InterfaceSoftwareAgent cannot find any subclass or instance related to *Non_Standard_Hardware_Problem*. Using rule 3, the InterfaceSoftwareAgent is required to gather and examine all properties, *hasSymptom*, *hasHardwareSymptom* and *isInstalledBy*, to determine if there is any direct connected classes from other category (categories other than problem types) contain the instances. Here, the direct connected classes are *Problem_Symptom*, *Hardware_Problem_Symptom* and *Installer*. The InterfaceSoftwareAgent ignores *Problem_Symptom* and *Hardware_Problem_Sympotm*, because they do not possess any property or instance. However, the InterfaceSoftwareAgent realizes that *Installer* has two instances. Thus, the two instances *Vendor* and *Help_Desk* are captured and displayed in the third drop box (rule 3a). Finally, the InterfaceSoftwareAgent activates the SolutionRetrievalAgent before terminates (rule 3a).

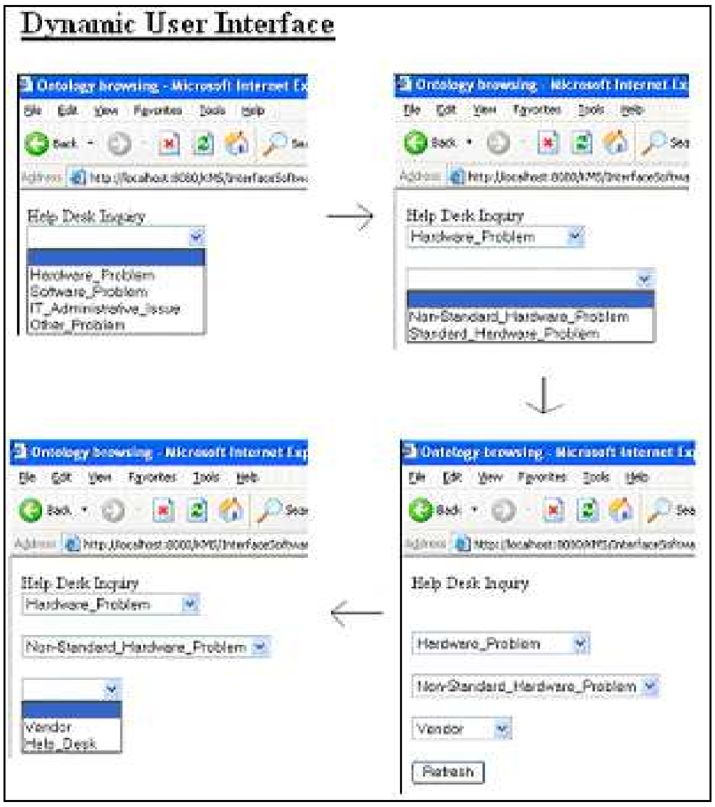

Dynamic User Interface
Ontology browsing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search
Address http://localhost:8080/KMS/InterfaceSoftwareAgent
Help Desk Inquiry
Hardware_Problem
Software_Problem
IT_Administrative_Issue
Other_Problem
Non-Standard_Hardware_Problem
Standard_Hardware_Problem
Vendor
Help_Desk
Refresh

Figure 6 Example to Demonstrate the Rule of the InterfaceSoftwareAgent (Dynamic User Interface View)

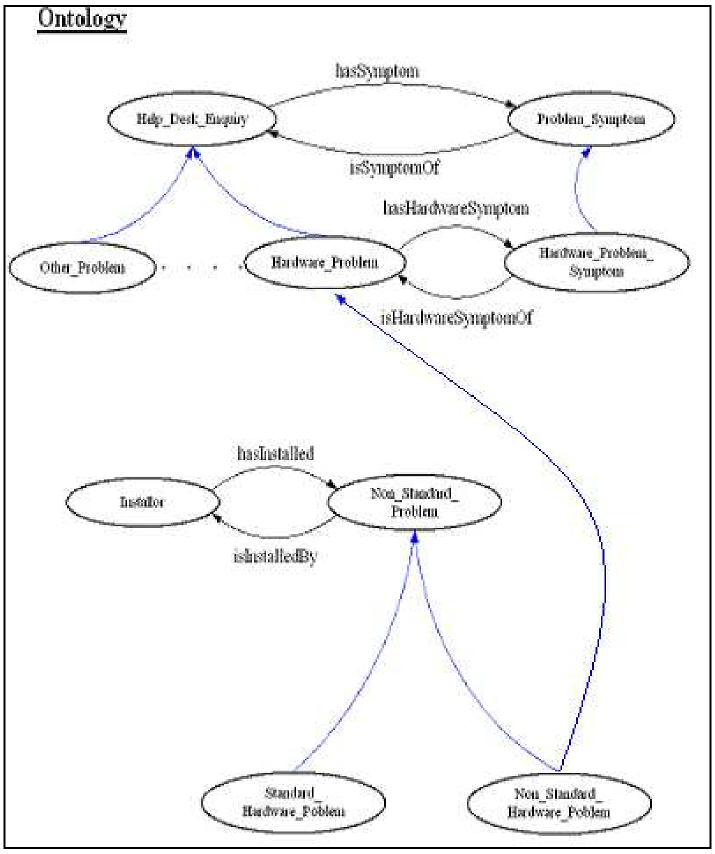
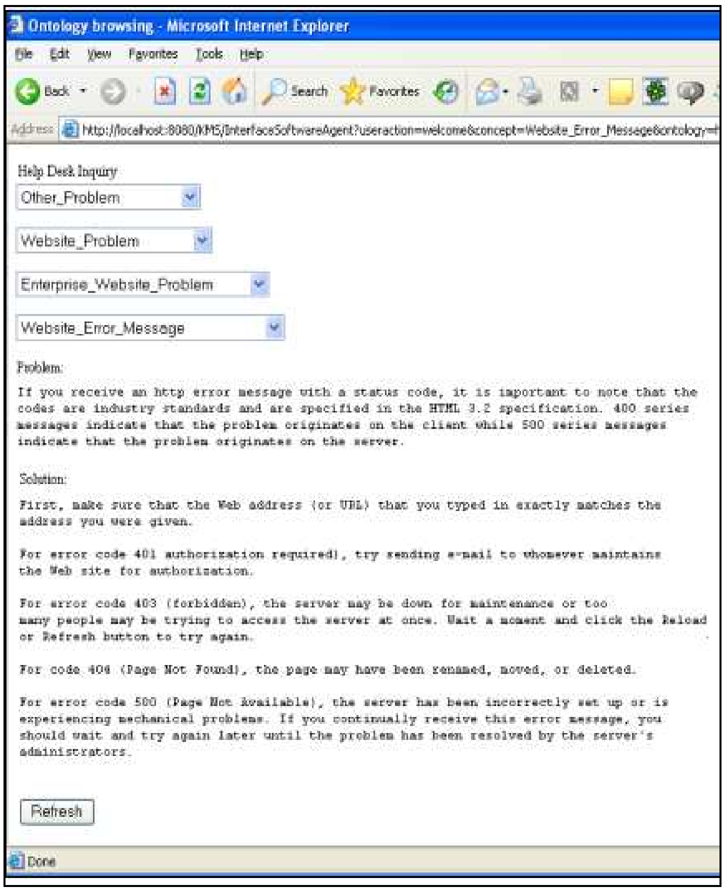


Figure 7 Example to Demonstrate the Rule of the InterfaceSoftwareAgent (Ontology View)

III. Illustration of Dynamic User Interface

To view the solution in the knowledge base, user is required to describe the problem types and their symptoms by choosing the related vocabularies from a series of drop boxes on the dynamic user interface. If there is a solution for the problem types, it will be displayed on the interface. Otherwise, a message will be shown to inform user that the solution for the chosen problem types and symptoms is currently unavailable.

To illustrate the functionalities of the user interface, let us consider two scenarios. In the first scenario, John gets an error message when he tries to access an internal website. He decides to search for solution in the online knowledge base. Firstly, he describes and identifies the problem types and symptoms by selecting *Other_Problem*, *Website_Problem*, *Enterprise_Website_Problem* and *Website_Error_Message* in four of the drop boxes. The dynamic user interface immediately retrieves the matching solution from the knowledge base and displays the solution (see Figure 8).


Ontology browsing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/KMS/InterfaceSoftwareAgent?useraction=welcome&concept=Website_Error_Message&ontology=
Help Desk Inquiry
Other_Problem
Website_Problem
Enterprise_Website_Problem
Website_Error_Message
Refresh
Done

Problem:

If you receive an http error message with a status code, it is important to note that the codes are industry standards and are specified in the HTML 3.2 specification. 400 series messages indicate that the problem originates on the client while 500 series messages indicate that the problem originates on the server.

Solution:

First, make sure that the Web address (or URL) that you typed in exactly matches the address you were given.

For error code 401 authorization required), try sending e-mail to whomever maintains the Web site for authorization.

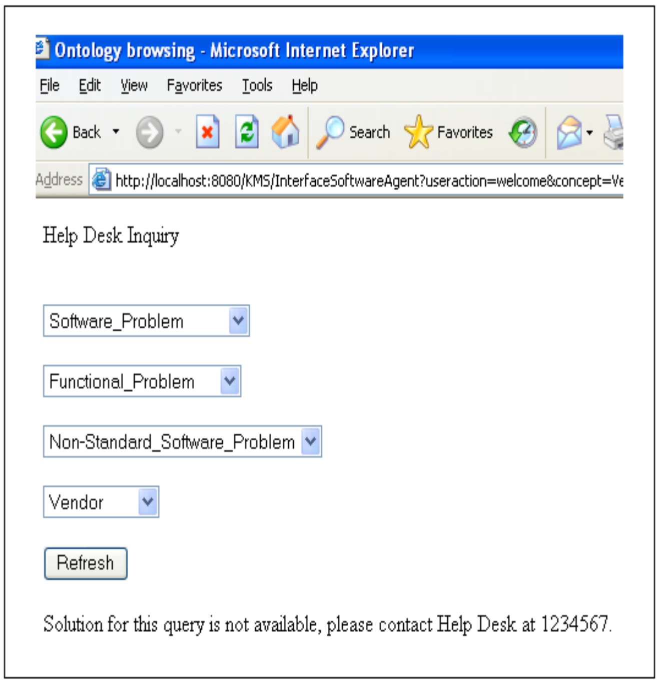
For error code 403 (forbidden), the server may be down for maintenance or too many people may be trying to access the server at once. Wait a moment and click the Reload or Refresh button to try again.

For code 404 (Page Not Found), the page may have been renamed, moved, or deleted.

For error code 500 (Page Not Available), the server has been incorrectly set up or is experiencing mechanical problems. If you continually receive this error message, you should wait and try again later until the problem has been resolved by the server's administrators.

Figure 8 Sample Screen of Retrieving Solution

In the second scenario, John has difficulties in using some of functions in SmartDraw installed by the vendor. Smartdraw is considered as a non-standard software in the company that he is currently working for. Thus, he decides to access the online knowledge base and search for a suitable solution. John identifies and describes the problem types and symptoms by selecting *Software_Problem*, Functional_Problem, *Non_Standard_Software_Problem* and *Vendor* in four of the drop boxes. As there is no matching solution stored in the knowledge base, a message is displayed to inform John that the solution is not available and he is asked to contact the help desk for assistance (see Figure 9).


Ontology browsing - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:8080/KMS/InterfaceSoftwareAgent?useraction=welcome&concept=Ve
Software_Problem
Functional_Problem
Non-Standard_Software_Problem
Vendor
Refresh

Help Desk Inquiry

Solution for this query is not available, please contact Help Desk at 1234567.

Figure 9 Sample Screen of Displaying “Knowledge Unavailable” Message

IV. Conclusion

The ontology is developed to represent various categories of problem types and their symptoms and to support the dynamic interface on which users can choose to describe and identify the problems. Apart from that, the popularity of using ontology to manage technical knowledge makes it possible for help desk to reuse other help desks or IT companies’ knowledge in terms of ontology. For example, company A has reached an agreement with Oracle and Norton to allow the help desk of company A to reuse technical support knowledge of Oracle and Norton products. By integrating or merging their ontologies, software agent from Company A should be able to retrieve technical knowledge from Oracle and Norton. This means that users in company A can make use of Oracle and Norton’s technical support knowledge to troubleshoot their own problems. The reusability of ontology also allows help desk to save a lot of resources and efforts in creating duplicate sets of knowledge that have already been created in other companies or help desks. The dynamic user interface is designed to pinpoint the weakness of the FAQ and “keyword search” interfaces used in online knowledge base. The interactive communication and easy-to-use capabilities of the dynamic user interface enable users to describe and identify their problems and the related symptoms in return for the most appropriate solutions.

References

[1] Antoniou, G. and Harmelen, F. *A Semantic Web Primer*. The MIT Press, London, England, 2004.

[2] Berners-Lee, T., Hendler, J. and Lassila, O. (2001) “The Semantic Web,” *Scientific American*. [Online] Available: http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2

[3] Broome, C. and Streitwieser, J. “What is E-support,” *Service and Support Handbook*, Help Desk Institute, 2002, pp.31-40.

[4] Central Computer and Telecommunications Agency. *IT Infrastructure Library: Help Desk*. HMSO Publication Centre, 1989.

[5] Gruber, T. and Olsen, G. "An Ontology for Engineering Mathematics," *in Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Streemann, Bonn, Germany, 1994.

[6] Heckman, R. and Guskey, A. "Sources of Customer Satisfaction and Dissatisfaction with Information Technology Help Desks," *Journal of Market Focused Management*, Number 3, 1998, pp.59–89.

[7] Klyne, G. and Carroll, J. (2004). Resource Description Framework (RDF) Concepts and Abstract Syntax. [Online] Available: http://www.w3.org/TR/rdf-concepts/

[8] Liu, H., Zeng, G. and Lin, Z. "A Construction Approach for Software Agents Using Components," *ACM SIGSOFT Software Engineering Notes*, Volume 24, Issue 3, 1999, pp.76-79.

[9] Marcella, R. and Middleton, I. "The Role of the Help Desk in the Strategic Management of Information Systems," *OCLC Systems and Services*, 12(4), 1996, pp.4-19.

[10] McGuinness, D. and Harmelen, F. (2004). OWL Web Ontology Language Overview. [Online] Available: http://www.w3.org/TR/owl-features/

[11] Nienaber, R. and Cloete, E. "A Software Agent Framework for the Support of Software Project Management," *in Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology*, 2003, pp.16-23.