Volume 1 No. 5, September 2011 ISSN-2223-4985
International Journal of Information and Communication Technology Research

http://www.esjournals.org

Automatically Detecting Relevant Technical Teams for Requested Issues in Issue Tracking Systems by Using Semantic Web Technology

[1]Mohamed Kholief, [2]Yasser Fouad, [3]Mohamed Hassan
[1]College of Computing and Information Technology,
Arab Academy for Science, Technology and Maritime Transport
Alexandria, Egypt
[2,3]Department of Mathematics and Computer Science, Alexandria University
Alexandria, Egypt

ABSTRACT

This paper describes the outcome research project – Automatic Team Detector (ATD) system - that has been carried out as a result of implementing the semantic web technology in issue tracking systems (ITSs) to enhance their functionality, performance and qualification. An automatic team detector (ATD) semantic web based application is developed to automatically assign - without any human support - the requested issues created in Issue Tracking System (ITS) to their appropriate technical support teams to handle and fix these issues. By using semantic web technology, semantic based tracking systems can be enhanced and improved. Hence, the main goal of the developed system - Automating Team Detector (ATD) semantic web based application - is to give the answers for the following questions:
To which team must the created issue in ITS be assigned? In the other words, who is going to be responsible for handing the created issue in ITS and doing the work on it until it is fixed?

Keywords: *issue tracking systems; semantic web; ontology development; automatic team detection*

I. INTRODUCTION

An Issue Tracking System (ITS) is a computer software web based application that allows customer and technical persons to record, follow the progress, manage, and maintain lists of identified issues. ITS allows developers, testers, technical support teams and the managers of an organization to keep tracking the progress of all requested issues and analyzes them, which is an important step to enhance organization system performance. Current ITSs support only keyword search. Some ITSs allow full keyword search for their reports, while others only support searching via a set of pre-defined filters (fields and values) applied to the entire issue database(8, 9). An issue tracking system needs hard efforts to make it more useful. This paper suggests a way to achieve this target.

Most of the used ITSs were developed by the organization itself to match their rules and requirements. After detailed study of current issue tracking systems such as Bugzilla, JIRA, Rational ClearQuest, Trac, … , weaknesses in assigned requested issues to their appropriate technical support teams have been found. Cases in point of these weaknesses are errors, extra effort, and excessive time consumption, and long life cycle until the requested issue arrives to its relevant technical support team. So, a semantic-based system is developed to tackle these issues. This system can automatically detect the appropriate technical support team that is responsible for dealing with the requested issue, depending on the issue's contained information and meaning by using semantic web technology. To realize this target, an ontology for the *information system (IT) team's job description and responsibilities domain*- which is described in section 5.1.1.1 - has been developed. Then, the automatic team detector (ATD) semantic-based application has been carried out. This application can analyze the created issue based on the semantics of its information, using the developed ontology. Hence, automatically identified - without any human support - the relevant technical support team is assigned the handling of the requested issue.

II. SEMANTIC WEB

Semantic Web was introduced in 2001 by Tim Berners-Lee as a vision of a new intelligent Web. In Berners-Lee's vision, data located somewhere on Web should be available, processable, and understood by both people and machines. The Semantic Web will add structure to the content of Web pages, being an extension of the current Web, in which information is given a well-defined meaning. The Semantic Web technologies in the form of ontologies and metadata are becoming increasingly prevalent and important (3, 2).

III. WHAT IS ONTOLOGY?

Ontologies are considered a major component of the Semantic Web. Ontologies provide a shared and common understanding of a domain that can be communicated between people and across application systems. An ontology consists of the various classes and properties that can be used to describe and represent concepts for a domain of knowledge. Classes represent concepts within a domain or across domains, and properties represent the relationships among these concepts. However, creating an ontology is very difficult, very time and effort-consuming, and requires a high

degree of expertise. As these ontologies grow in size, they become more and more difficult to create, use, understand, maintain, transform and classify. There is no standard steps for creating an ontology for a specific domain (3, 4, 5). The most widely used and best suitable definition of an ontology for our purposes is:

"An ontology is a *formal, explicit specification* of a *shared conceptualization*". (Gruber 1993) (5, 7)
Formal implies that an ontology should be machine-readable. However, an ontology is not "active;" it cannot be run as a program. It represents declaratively some knowledge to be used by programs. ***Explicit*** means that the type of used concepts and the constraints on their use are explicitly defined. ***Specification*** means a formal and declarative representation. In the data structure representing the ontology, the type of concepts used and the constraints on their use are stated declaratively, explicitly, using a formal language. ***Shared*** reflects that an ontology captures consensual knowledge. That is, it is not related to a specific individual, but accepted by a group. ***Conceptualization*** refers to an abstract model, a simplified view of the world. If the knowledge base of an intelligent system is to represent the world for some purpose, then it must be committed to some conceptualization, explicitly or implicitly. That is, every body of formally represented knowledge is based on a conceptualization which is based on the concepts, objects, and other entities that are assumed to exist in an area of interest, and the relationships that exist among them. This explains the meaning of the term "world"—in practice, "world" actually refers to some phenomenon in the world, or to some topic (or topics), or to some subject area (4, 5 ,7).

A. Main components of an ontology

Basically, any ontology should contain:

i. **Classes** (sometimes called concepts); Classes are the focus of most ontologies. Classes represent concepts in the domain of discourse (either physical/specific or abstract/conceptual). They could be organized in taxonomies to define superclass - subclass hierarchy. Subclasses represent concepts that are more specific than the superclass.
ii. **Properties** (sometimes called attributes, roles or slots); Properties represent association between concepts. They describe various features and attributes of each concept and instance. They are usually binary.
iii. **Facets** (sometimes called Restrictions on slots or role restrictions); Facet is used to represent information about properties .Slots can have different facets describing the slot value type, allowed values, the number of the values, and other features of the values the slot can take.
iv. In**stances** represent elements or individuals in an ontology. An ontology with a set of individual **instances** of classes constitutes a **knowledge base**. In reality, there is a fine line where the ontology ends and the knowledge base begins (6, 7).

IV. ISSUE TRACKING SYSTEMS (ITSS)

Improve the quality, upgrade, troubleshooting, fixing defects, replacing and reduce expenses are a constant feature for any system to provide better and greater functionality, however, those issues that can be done for the system must be controlled and properly handled(?, ?, 11). There are many software applications that have been developed oriented around controlling and tracing the lifecycle of bugs, enhancement, Problems, tasks, and requirements, those applications called Issue tracking systems (ITSs). Issue tracking systems (ITSs) are considered as a useful way for controlling, tracking and offering large archives of issues as a knowledge base which is helpful in troubleshooting advice(10, 12, 13).

Change request Workflow (Also known as issue life cycle)

A change request is a document contained within all available initial information about the requested issue. Its type states what needs to be accomplished, etc. but leaves out how the request should be carried out. Change requests are commonly created by customers, the help desk, Operators or customer support call center staff through the ITS application (16). An example scenario is presented to demonstrate how a common issue tracking system would work:

The life cycle of a change request starts as it is opened when a Customer support team member (also known as help disk employee) receives a telephone call, email, or other form of communication from a customer about a requested issue. After receiving a new request ticket, the first step is verification which means that the customer support employee verifies that the requested issue is a real issue and also ensures that enough information about this issue is obtained from the customer. This information generally includes the environment of the customer, when and how the issue occurs, and all other relevant details. The customer support employee creates the issue in the ITS system, entering all relevant data, as provided by the customer. The issue is entered in the ITS system using a simple form and placed into one of a few categories (depending on its type, which part of the system the issue relates to, etc). Here, the ticket is assigned to the analyst who is responsible for receiving the created issue and analysing its information and then re-assigning it to the relevant technical support team in charge of handling and fixing this issue. A specialist accepts the request ticket as it is assigned to him, indicating that he has seen it and that he is aware that it is his responsibility to resolve and handle it (10, 13, 14, 15). From the previous explanation of the issue life cycle we can conclude that assigning requested issues to their responsible technical team to handle them can be considered as an inefficient process due to the extra time consumption, errors, the long path

taken by the requested issue to reach the responsible technical team. In addition, issues are waiting in a queue until they are assigned by analyst team. Finding a solution for handling this weakness of assigning requested issue to the responsible technical team in simple, accessible, trusted and short way is the main goal of our work.

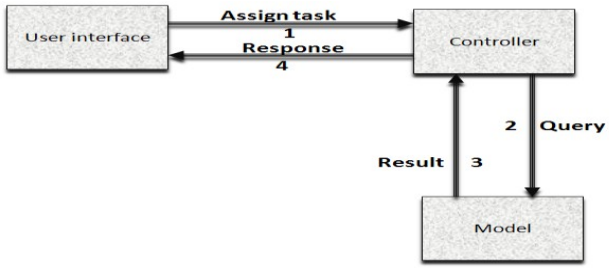
V. The Automatic Team Detector (ATD) Semantic Web Based Application

The Automatic Team Detector (ATD) semantic web based application receives the created issue, and analyzes the issue's information, depending on the semantics of this information, using the developed ontology for the *information system, team's job description, and responsibilities domain.* The appropriate technical support team responsible for handling this requested issue is then automatically detected.

A. ATD (automatic team detector) Application Architecture

Figure 1 shows the simple architecture of ATD (Automatic Team Detector) semantic web based application.

Figure1: Automatic Team Detector (ATD)



Application Architecture

1) Model Layer (Knowledge Representation)

The following ontology is represented in RDF and OWL full format languages: the developed ontology for the *information system (IT) team's job description and responsibilities domain*,

In this layer the SPARQL code retrieves the needed information about the team that is responsible for handling the requested issue from the developed ontology and returns by the result to the controller layer.

The ontology for the *information system (IT) team's job description and responsibilities domain is c*onstructed by finding and gathering the terms, concepts, relationships, and concept hierarchies existing in large collections of unstructured text documents. These documents give information and a detailed description of the job and the responsibility for each team in the information system (IT) sector. Steps that have been followed to develop the ontology for *information system (IT) team's job description and responsibilities domain* are listed in the following sections based on criteria in (5,6,7).

1. The domain and scope of the ontology

The domain and scope for the developed ontology is determined by answering a set of questions such as:

What will the domain of the ontology cover?
ANS: The domain will cover the job description and responsibilities for each team in the information technology (IT) sector.

For what purposes will the developed ontology be used?
ANS: The ontology will be used to determine the IT team that is responsible for handling the created issue.

What type of questions should the information in the ontology answer?

ANS: it should answer questions such as: what is the IT team that is responsible for tackling and solving the requested issue?

Who will use and maintain the ontology?
ANS: All members from inside the organization and customers from outside the organization will use it. The system administrator will maintain it.

Enumerating the important terms in the ontology

Information about characteristics, job description and responsibilities for each team in the IT sector should be collected and analyzed. This information has to be documented in structure format documentation. The relationship between all IT sector teams should then be implemented.

By answering the following questions, all terms that will be implemented in our ontology are determined:

What are the terms we would like to talk about?
ANS: Terms that we would like to make statements about or to explain are:
IT sector, Administration Team, Application team, Developer Team, Closed System, Open System, DB2 Administration Team, DB2 Developer Team, database administration Team, database developer Team, database Objects, database Operation, database Problem, database Responsibilities, database Software, database Team, Developer Responsibilities, Developer Software, table, index, Column, row, IP … and so on

What properties do these terms have?
ANS: Properties that these terms have are:-
Analysis, application, patches, assignment, maintenance, installation, solution, configuration, documentation, development, elimination, implementation, Improvement, performance … and so on

What do these terms include?
ANS: These terms include:

- data about all IT sector teams such as network team, database team, operation team, developer team, etc ,
- Information about the team such as name, IP phone, location, position title, etc.
- description about all issues (problems, tasks, bugs, etc) that every team can deal with,
- Different types of problems such as network, database, all software, and hardware problems, etc.
- Subtypes of software such as OS software, network software, and so on.

2. Defining the classes and the class hierarchy

In this step we use several possible approaches in developing a class hierarchy:

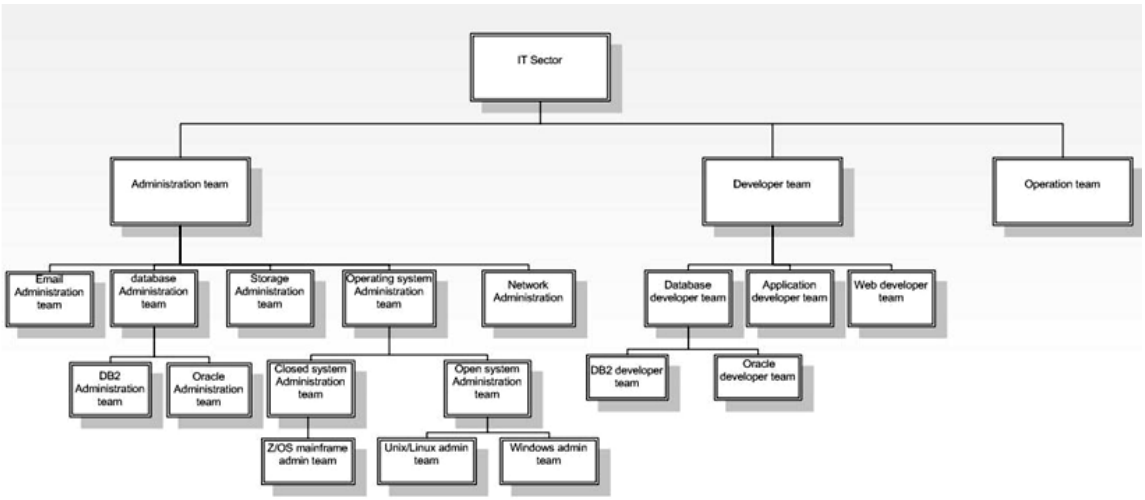


Figure 2: The relation between all IT sector teams classes (class & subclass relation)

- A **top-down** development process starts with the definition of the most general concepts in the information technology, *information system (IT), team's job description, responsibilities domain,* and the subsequent specialization of the concepts.

For example, we start by creating classes for the general concepts such as *IT team, responsibilities, bugs, tasks, problems, hardware, software,* etc. We specialize IT team class, for instance, by creating some of its subclasses as administration team, developer team, operation team, etc. The developer team class can for example be further developed into database developer team, application developer team and web developer team, etc. Figure 3 shows part of *IT team* hierarchy.

Figure 3: A bottom-up development process for DB2 admin team and oracle admin. They are the most specific classes in the hierarchy (or the bottom level concepts).

- A **bottom-up** development process starts with the definition of the most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts.

For example, we start by defining classes for DB2 Administration team and Oracle Administration team. We then create a common superclass for these two classes—database administration team—which in turn is a subclass of Administration Team.

- A **combination** development process is a combination of the top-down and bottom-up approaches: We define the more salient concepts first and then generalize and specialize them appropriately. For example, we start with a few top-level concepts such as *IT team* and a few specific concepts such as *application developer team*. We can then relate them to a middle-level concept such as *developer team*.

3. Defining the properties of classes (slots)

The classes alone will not provide enough information to answer the questions raised in step 1. So, once some of the classes are defined; the internal structure of concepts is described. Classes from the list of terms, created in Step ш, are selected 0. Most of the remaining terms are likely to be properties of these classes. These terms include, for example, *responsible for, take backup, solve, maintain, has name, has IP phone, has position*, etc.

4. Defining the facets of the slots

Slots can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values that the slot can take. For example, the value of a *name* slot (as in "the name of a member") is one string. That is, *name* is a slot with value type String. A slot *has Location* and it can have multiple values which are instances of the class *Location*. Examples of defined Slot-value type are:

- **String** is the simplest value type which is used for slots such as *name*. Hence, the value is a simple string

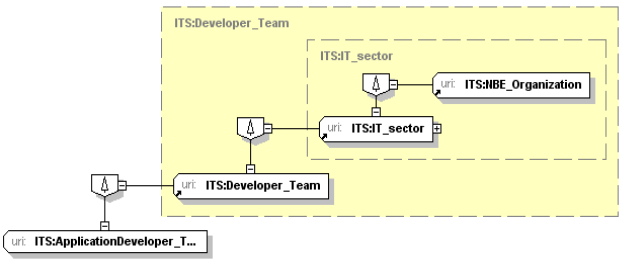


Figure 4: The different levels of the team taxonomy: *NBE Organization* is the most general concept

- **Number** describes slots with numeric values. For example, a *NumberOfMembers* slot of *IT_Team* class can have a value type Integer.
- **Enumerated** slots identify a list of specific allowed values for the slot. For example, it can be specified that the *Locations* slot will take one of the three possible values: *Shrief_site*, *Smart_Village*, *World_Trade* and *Plaza_Site*. In Altova Semantic Works 2010 the enumerated slots are of type *one*.
- **Instance**-type slots allow definition of relationships between individuals. Slots with value type Instance must also define a list of allowed classes from which the instances can come. For example, a slot *hasResponsibility* for the class *IT_Team* may have instances of the class *Responsibilities* as its values.

Figure 4 presents an example of slot definition. String data type is the value type of the *hasName* slot and The *IT_Team* class is the domain.

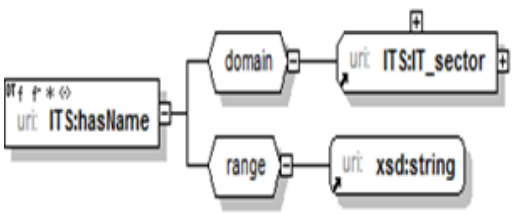


Figure 5: Definition of the slot *has Name* at the class *IT Team*.

5. Creating instances

The last step is to create individual instances of classes in the hierarchy. Defining an individual instance of a class requires:

(1) Choosing a class,
(2) Creating an individual instance of that class, and
(3) Filling in the slot values.

For example, an individual instance *OracleDBA* represents a specific type of Oracle DBA team. *OracleDBA* is an instance of the class *IT_Team,* representing all members in oracle DBA team. This instance has the following slot values defined:

- ITS:*hasLocation:* instance of *location* class
- ITS:*hasMembers:* instance of *Members* class
- ITS:*hasName: Oracle DBA*
- ITS:hasResponsibility : instance of *Responsibilities* class
- ITS:hasPosition: instance of *Positions* class

5.3 Controller Layer (Data Extractor)

This layer is the java application. It is responsible for receiving the requested issue from the interface layer and it sends it to the Model Layer. Then it *sends* the response back to the interface layer. This response about the team's information is responsible for dealing with the requested issue.

5.4 User interface Layer

This Layer is the top layer of the application where the web pages are created and the requests are sent from. The home page of the application has a text editor area in which the query - requested issue – is entered and submitted by clicking the query button.

VI. Conclusion

The Automatic Team Detector (ATD) application can receive the created issue and analyze it based on the semantic of the issue, using the developed ontology for the *information system (IT) team's job description and responsibilities domain.* Then, the created issue will be identified and assigned to the appropriate technical support team that is responsible for handing it., this process has numerous privileges such as:

- Time saving due to the quick life cycle of the issue from the creation until it becomes handled.
- Cost reduction; there is no need to help desk team.
- No errors in assigned issues to appropriate technical team; there are no human mistakes in assigned issues process because the ATD

application will automatically detect the appropriate technical team to those requested issues.
- Effort saving; there is no need for wasted effort by help desk team in assigned issues to the related technical team.
- Fast response; which results in impressing and satisfying customers as the requested issue will be sent to the technical team in a short time, easy and trusted way.

REFERENCES

[1] Berners-Lee T., Hendler J. and Lassila O., 2001.*The Semantic Web. The Scientific American, vol. May17, 2001. Retrieved from* http://www.scientificamerican.com/2001/0501issue/0501berbers-lee.html

[2] Antoniou G. and van Harmelen F., 2008. *A Semantic Web Primer*. Second Edition. The MIT Press, Cambridge, Massachusets, London, England.

[3] Segaran T., Evans C. and Taylor J., 2009. *Programming the Semantic Web*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[4] Berlin Heidelberg V., 2009. *Model Driven Engineering and Ontology Development*. Springer Dordrecht Heidelberg London New York.

[5] Natalya F. Noy and Deborah L. McGuinness,2001. O*ntology Development 101: A Guide to Creating Your First Ontology. Knowledge Systems Laboratory*, Stanford University.

[6] Berlin Heidelberg V., 2006. *Ontologies for Software Engineering and Software Technology,* Coral Calero · Francisco Ruiz ·Mario Piattini (Eds.).

[7] Bermejo J., 2007, *a Simplified Guide to Create an Ontology*. Madrid University.

[8] Hauner A., 2008. *Bug tracking pro vývojáře*. Retrieved from SoftEU s.r.o. Blog: http://blog.softeu.cz/prednasky/2008/prednaska-bug-tracking-pro-vyvojare/

[9] *Wikipedia. Issue Tracking System*. Retrieved from http://en.wikipedia.org/wiki/Issue_tracking_system#Architecture

[10] Blair S., 2004. *A Guide to Evaluating a Bug Tracking System*. Retrieved from Scribd.com: http://www.scribd.com/doc/7046090/A-Guide-to-Evaluating-a-Bug-Tracking-System

[11] Kaner, C., 2002. *Bug Advocacy*. Retrieved from Cam Kaner Home Page: http://www.kaner.com/pdfs/BugAdvocacy.pdf

[12] Atlassian Pty Ltd., 2008. I*ssue Types*. Retrieved from JIRA - Local help: http://jira.atlassian.com/secure/ShowConstantsHelp.jspa?decorator=popup

[13] Hauner A., 2008. *Hlásíme chyby v software*. Retrieved from SoftEU s.r.o. Blog: http://blog.softeu.cz/prednasky/2008/prednaska-hlasime-chyby-v-software/

[14] Birlasoft Ltd., 2008. *Bug Life Cycle*. Retrieved from Scribd.com: http://www.scribd.com/doc/7289591/Bug-Life-Cycle

[15] *An issue's life cycle.* Retrieved from http://www.openoffice.org/nonav/scdocs/issue_lifecycle.html

[16] *Wikipedia*. Change Request. Retrieved from http://en.wikipedia.org/wiki/Change_request