ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol.36, No.1

NATURAL LANGUAGE AS PROGRAMMING PARADIGM IN DATA EXPLORATION DOMAIN

Algirdas Laukaitis, Olegas Vasilecas

Fundamental Sciences Faculty, Vilnius Gediminas Technical University
Saulėtekio al. 11, LT-10223 Vilnius, Lithuania

Abstract. In this paper we present the progress of the natural language usage as the programming paradigm for information extraction in distributed database environments. Personal assistants form an environment where distributed knowledge is explored with the JMining interlingua language to support communication between the mobile agents, natural language queries and the mobile agents working environment servers. The Aglets framework is used to build mobile agents and test conceptual designs for information gathering. The implementation of the prototypes using the aglet framework shows that even with the state of the art natural language technologies the applications development is achievable only on the narrow domain and with the small interlingua language design. Presented architecture can be integrated as a part of a corporate information delivery Web portal to bring new modalities for user interfaces with the possibility to share locally stored knowledge bases.

Keywords: Natural language as programming paradigm, personal assistants architecture, mobile agents, aglets, information extraction in distributed database environments.

1. Introduction

Data environments are becoming more and more complex as the amount of information a company manages continues to grow. Information delivery web portals have emerged as the preferred way to bring together information resources. Using information delivery web portal, your organization's employees, customers, suppliers, business partners, and other interested parties can have a customized, integrated, personalized, and secure view of all information with which they need to interact. But one big challenge remains for organization: how to teach employees or customers to use and understand complex database environment without involving experts and IT resources which are costly and time consuming. Natural language interfaces between human and programmable agent can be the answer.

From the early 80's and 90's there was many efforts involved in the research of natural language use for information extraction from data base management systems (DBMS) [3], [8]. Natural language database interfaces (NLDBIS) are systems that allow users to access information stored in a database by formulating requests in natural language. The system that supports (NLDBIS) functionality automatically would translate user sentences to adequate SQL script, query some DBMS and return results to the user. NLDBIS have received particular attention within the natural language processing community (see [2] for reviews of the field), and they constitute one of the first areas of natural language technology that have given rise to commercial applications. Some successes have been achieved and some commercial applications emerged but the NLP techniques have not become a popular approach for DBMS interfaces. As was mentioned by many researchers [7], [28], [33] this is due to:

1. Graphical and menu driven interfaces achieved the level of sophistication that many data analyst can do analysis without deep knowledge of some data queering language (e.g. SQL), on the other side NLP techniques has not been able to deliver interfaces of adequate sophistication.
2. Most research and achieved results report on the possibility to generate only one data queering script (in most cases this was one SQL sentence) generated from one natural language sentence. They do not support complex dialog, which is the most usual case in real life when we want interactively to build adequate request.
3. Most systems are commercial products [22]. They are close systems and there are difficulties in extending such systems.
4. In most available systems only system administrators are able to parameterise the system. There

are no available systems in which learning process is integrated in the user daily work. The resent advances is building personal assistants for such fields like an adaptive information gathering from the internet [5] or personalized learning knowledge maps [24] will renew interest in (NLDBIS) field.

In this paper several paradigms are suggested for solving problems mentioned above.

1. The dialog management instead of one sentence must be provided for the communication with the information gathering agents. You can get really frustrated with the answers from the system constantly suggesting reformulate query statements. Several studies show that software engineers spend half of their time communicating in order to get information. We can hardly expect the at this stage of natural language technological development computers will be more intuitive than software engineers in understanding requirements presented by natural language.
2. Some philosophical arguments in the field of the information systems suggest that most information systems are developed as "trivial machines" to be predictable and controllable. Requirements for more flexibility and adaptability from the information systems can lead to the use of systems based on intelligent, autonomous and mobile agents. Nevertheless, mobile agents paradigm is struggling to find the way out of research area into the sound industrial applications. In this research we used mobile agents to gather distributed metadata, query distributed databases and interact with several humans to get more information. Behavior unpredictability that emerged from such architecture is one of the exciting challenges we met in our experiments.
3. An interlingua language employment demonstrated promising results in many areas of language processing and particular in the area of machine translation. In this research we redesigned the JMining language, which was primary, used by non-programmers to build small, reliable web based information extraction programs. The new language was used by the agents to communicate with the JMining servers, that they visited during information gathering session. We assume that each server visited by mobile agent has natural language interface to communicate with humans.

The contribution of this paper is threefold: Firstly, we introduce architecture of NL dialog for information delivery web portals. Proposed architecture is characterized by its flexibility to extend and a possibility to build complex information delivery web portals. Secondly, we investigate distributed agents architecture where each agent move their code and data to remote hosts and locally solves adequate tasks and returns to their mother host with the solutions. Thirdly, all presented concepts are implemented as Java open source project.

The remaining sections of this paper are organized as follows. Section 2 presents the JMining portal architecture and JMining natural language support infrastructure. In Section 3, we present information delivery portal infrastructure. In Section 4 the dialog-supporting agent is presented and dialog supporting modules are presented in Section 5. Final section concludes the paper.

2. General architecture

To tackle mentioned problems the distributed heavy personal assistants (DHPA) architecture is suggested and implemented. The word "heavy" is used in the sense that each personal assistant (PA) has almost all components of enterprise application. Figure 1 shows the main components of the JMining personal assistant. On the left side of the figure is the general view of the infrastructure. We assume that all conceptual modules are installed on each computer, which is involved in the information extraction process. The grey shaded blocks represent the modules that have been originally built by the authors of the paper and they are discussed in the following subsections.

The white blocks represent software modules that are developed by other developers. All of them are free of charge (an important issuer if we are planning to install them on all working personal computers) and the following example shows our configuration tested in the experiments.

1. ***Database management systems***, like MySQL, can be installed on all personal computers without bearing any additional cost for the company.
2. We used ***Tomcat servlets container*** as the Web server. Again it is free of charge end it is used to handle user interfaces.
3. Desktop search module represents software like ***Google desktop search***. Its primary use in our architecture is to faster documents search for the stationary and mobile agents. Our mobile agents by imitating Web browser used desktop search engine to find relevant documents for handling users queries.
4. ***Knowledge management*** represents software like Protégé and is used to manage local knowledge files.
5. ***Mobile agent server*** is used to coordinate the mobile agents activities and communications.

In our experiments we used Aglets framework for mobiles agents implementation and coordination.

All those modules are parts of personal assistant and in the JMining architecture we separate two kinds of personal assistants: 1) local personal assistant is responsible for handling local knowledge base of one particular employee 2) enterprise personal assistants are responsible for handling centralized knowledge bases and by general means they represent enterprise applications. Both, local and enterprise personal assistants have the same conceptual architecture, the differences are in the hardware profiles: local personal assistant is assumed to run on the single powerful PC computer and the enterprise personal assistant uses enterprise level hardware profile.

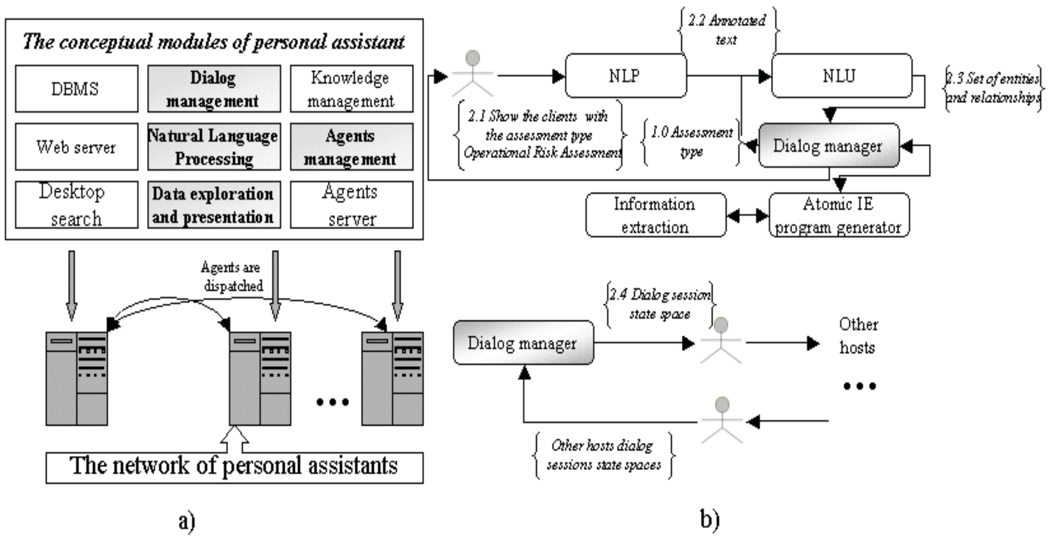
The conceptual modules of personal assistant
DBMS
Dialog management
Knowledge management
Web server
Natural Language Processing
Agents management
Desktop search
Data exploration and presentation
Agents server
Agents are dispatched
The network of personal assistants
a)
2.2 Annotated text
NLP
NLU
2.3 Set of entities and relationships
2.1 Show the clients with the assessment type Operational Risk Assessment
1.0 Assessment type
Dialog manager
Information extraction
Atomic IE program generator
2.4 Dialog session state space
Dialog manager
Other hosts
Other hosts dialog sessions state spaces
b)

Figure 1. a) conceptual modules of the personal assistant, b) general communication processes between human, JMining personal assistant and information gathering personal assistants

Next, we will discuss the four conceptual modules we developed to support natural language interface for developing applications that extract information from databases and presents it to the user.

Dialog management – represents state space dialog management agent. The state space dialogue strategy is a mapping from a set of states (which summarize the entire dialogue) to a set of actions (such as identification of tables and database queries). The state space is defined by the collection of all variables that characterize the state of the dialogue system at a certain point in time. To avoid combinatorial explosion the designer of the system must consider how on the one hand to limit number of variables and the number of values assigned to variables and on the other hand how to use enough variables so that to cover particular domain with various dialog flow possible paths. The set of actions describes what the system can do, i.e. the set of functions the system can invoke at any time (e.g. play a certain prompt, query a database, hang up, etc.). The strategy is a mapping between the state space and the action set. For any possible state the strategy prescribes what the next action to perform is. As a result of the action and its interaction with the external environment (e.g. user, database, etc.) the system gets some new observations (e.g. database entities, attributes, etc.) and they can modify the state of the system. This process continues until a final state is reached (e.g. the state with legitimate SQL, XML script) [20].

Data exploration and presentation objects are the domains of our investigations. It is a collection of objects for queering corporate databases, analyzing retrieved data and presenting results to the user in graphical and textual templates.

Natural language processing is an implementation of morphology, syntax and lexical semantics analysis of user sentences presented to the system in the form of natural language.

Agents management module is used to query other personal computers preinstalled with the same framework. The use of mobile agents in architecture is reasoned by the approach, which argues that knowledge consists largely of a personal, stored locally data files. Mobile agents can travel to various hosts where local knowledge is stored and gather necessary information that meets user request. The paradigm of agents is a very promising approach to overcome some of the problems connected with heterogeneity on the side of the data sources as well as on the side of the users. As agents should operate autonomously and can be loosely coupled, they are well suited for the integration of distributed heterogeneous data sources, building unifying wrappers around them. This becomes especially beneficial, if agents can learn to extract information from an information source automatically (see for example [10] and [25]). On the side of the users, the paradigm of personal information agents offers a way to encapsulate the interests, the knowledge as well as the preferences of individual users. Personal agents can take the role of mediators between users and information sources, as well as between users among each other (see also [10] and [30]). Furthermore we present an agent architecture consisting of a set of asynchronously operating agents. This architecture enables us to perform sophisticated data and interaction analysis, without loosing the property of short respond times essential for interactive work in real-time. Based on the paradigm of mobile agents, we present a model for expressing knowledge that has been acquired continuously by individuals and groups of users and for using this as a means for semantic identification of various elements to build necessary web applications.

3. Natural language based information delivery portal

There are many commercially successful information delivery web portal products that are available in the market and the open source community is catching up. For example, the open source information delivery portal JMining [15], [16] has all conceptual functionality that is provided by its commercial counterparties like SAS [29], Oracle [26], Microsoft [23], Information builders [13], etc. Indeed, proprietary products are close systems without almost any possibility to adapt them for the natural language interfaces. Another important issue in the information systems development is the scale of the complexity that faces us when we are trying to use natural language. To be able move from SQL paradigm to application are even information system development paradigm by natural language we need establish well structured architecture where each element of the architecture can be manipulated by the state-of-the-art natural language technologies. Figure 2 shows the main components that we integrated with the natural language interface.

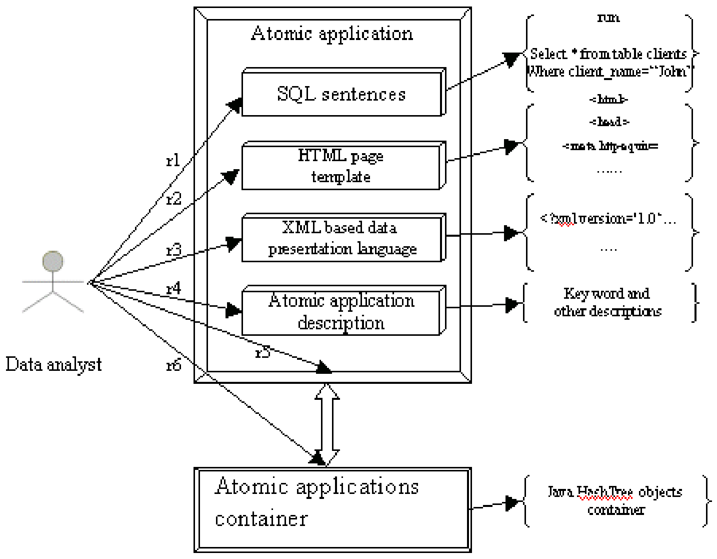
Atomic application
SQL sentences
HTML page template
XML based data presentation language
Atomic application description
Data analyst
r1
r2
r3
r4
r5
r6
Atomic applications container
run
Select * from table clients Where client_name="John"
<html>
<head>
<meta http-equiv=
......
<?xml version='1.0'...
....
Key word and other descriptions
Java HashTree objects container

Figure 2. Idea of the atomic application

As mentioned above, one of the biggest problems with NL dialog systems is the number states. Reduction of this number is one of key problems in any dialog-based system. It is why we used JMining IDP because a fundamental idea behind this IDP is that its architecture is based on atomic applications container.

IDP JMining is implemented as database and platform independent. Data base system accessed by one of the following protocols (ODBC, JDBC or XML). The JMining is server-based application written completely in Sun's Java programming language. Because the Jmining modules are written in Java, they can run on any server platform that supports a Java Virtual Machine. Data used by the portal: account credentials, access controls, demographics, personalization parameters, and configuration information can be stored within an X500 directory services database accessible through LDAP (Light-weight Directory Access Protocol). All those data set can be stored into metadata storage of our dialog management system and then accessed and manipulated by other system of the dialog management. By such approach we achieve that such users as system administrators can manipulate (retrieve, modify or create new) some objects stored within LDAP server during NL conversation with the system. Next we describe mentioned fundamental idea of used IDP, which is call atomic applications container.

By ***atomic application*** (Figure 2) we understand the small web application, which contains the following components: database script, user interface HTML page, data representation script (XML, XSL, etc.) and documentation page (additionally there is connection to DBMS parameters, name of the application, and parent name of the application to organise all atomic applications in one single directory structure). Atomic application structure in some way resemblance to well knows web applications developing technologies like Servelets, JavaServer Pages (JSP) and Active Server Pages (ASP). With such technologies like JSP you can have the full power of general programming language like Java. But on the other hand it is unlikely that nonprogrammer or person without Java knowledge can handle such technology. On the other hand by putting more constraint on the web applications structure we achieved that nonprogrammer can successfully develop web applications. Surely that doesn't mean that no IT skills required. The user of this IDP software actually is the user who previously used such products like Microsoft Access to develop some local based database applications. Such user mostly has a good understanding of a database model as well as some basic SQL knowledge (sure most often that is no need for the user to write SQL sentences, instead it is done by interactive software wizards).

Atomic application represents one of the basic classes. Object derived form the class (like a brick in the house) is used to build an enterprise information delivery web solution. As mentioned above the set of such atomic applications can bring full portal solution to some business subject. We think that the small number of components that can be manipulated to build reliable small web application is attractive feature for the systems number of control variables is a big constrain. Below we describe in details these components that can be manipulated by our dialog management system.

SQL – set of SQL statements that are send to DBMS. There unlimited number of SQL statements that can be send to SQL server within one request but the last one must be SELECT type SQL statement. These statements are then executed in the selected database management system to retrieve information and to display it to the user through selected reporting template, which can have graphical or textual formats. Also the users have the choice of modifying these SQL statements as well as reporting templates to create their own applications.

HTML page – HTML document used to set user request parameters which can be used later to form dynamic SQL statements. Even if the primary intention of this parameter was to support dynamic SQL statements, it can be used as an independent HTML page for other web portal need. User has choice to keep parameter values permanently to the end of Internet session or just to the end of request implementation by web server.

Type of visualization object – used to choose selected data representation object from web server (e.g., graphic, bar char, some form of text (XML, HTML, TXT) layout, etc.).

XML (XSL) – Extensible Markup Language (XML) [34], [35] offers its users many advantages, including: simplicity, extensibility, and openness. XML as the atomic application component is used as some script for data visualisation (e.g., it can say which column forms x or y axis in a graphic or which field represents grouping, total variables and how they must be presented in the HTML document, etc.). From DBMS selected data are parsed with statements that are extracted from XML document. If the data comes from XML document (it is common situation in organizations that some data now can be received from XML documents instead traditional of DBMS) document can be used to transform data to HTML format.

The proposed structure of atomic application is optimal in the following way: it contains the minimum number of components that are required for building complex web portal. This IDP architecture is robust to some faults done by non-professional programmers (bugs can effect only one atomic application but the whole system is unaffected).

One exclusive property of proposed architecture is that the whole development and deployment is done only through web browser interface. Developers have a huge feasibility and mobility by choice of the platform. Security level is the same as in most web based e-commerce applications.

4. Natural language processing for scripts generation and agent's coordination.

Figure 3 presents the basic steps we use in natural language processing and understanding. The final result of those steps is identified triplet: *Entities, Relationships and associated probabilities.*

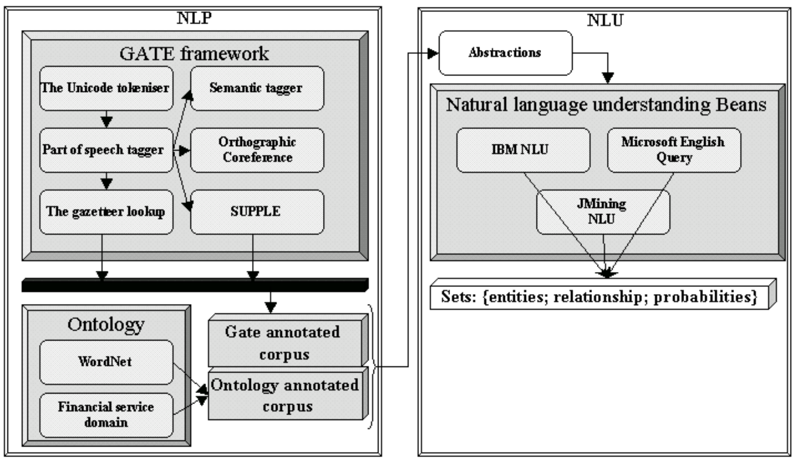

Figure 3. Basic steps in natural language understanding

The keystone of the whole process is *GATE – Natural Language Processing Engine.* GATE – General Architecture for Text Engineering – is a well-established infrastructure for customisation and development of NLP components [4]. It is a robust and scalable infrastructure for NLP and allows users to use various modules of NLP as the plugging. We briefly describe modules used in our research for building concepts vector spaces. *The Unicode tokeniser* splits the text into simple tokens and is used for the next steps of the natural language processing. *The tagger* is a modified version of the Brill tagger, which produces a part-of-speech tag as an annotation on each word or symbol. The list of tags can be found in [4]. We used it to extract nouns and verbs and remove all other words from the dictionary. *The gazetteer* further reduces dimensionality of the documents corpus prior to classification. It uses the lists of named entities and annotates text with class labels such as cities, organisations, days of the week, etc. We replaced each named entity with the label of the class. *Semantic tagger* – provides finite state transduction over annotations based on regular expressions. It produced additional set of named entities and we replaced each named entity with the class label. *Orthographic Coreference* – the module adds identity relations between named entities found by the semantic tagger. Reduction of the state space dimensionality is achieved by replacing marked tokens with named entities class labels found by the semantic tagger. *SUPPLE* is a bottom-up parser that constructs syntax trees and logical forms for English sentences. We used it only to remove tokens not annotated by this module. All modules within the GATE produced annotations – pairs of nodes pointing to positions inside the document content, and a set of attribute-values, encoding linguistic information.

We are finishing this section by presenting our motivation of using mobile agents approach. Mobile agents are computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming 'back home' having performed the duties set by its user. Mobile agents may cooperate or communicate by one agent making the location of some of its internal objects and methods known to other agents. By doing this, an agent exchanges data or information with other agents without necessarily giving all its information away [1].

The mobile agents need *not* be stationary; indeed, the idea is that there are significant benefits to be accrued, in certain applications, by putting away static agents in favour of their mobile counterparts. These benefits are largely *non-functional*, i.e. we could do without mobile agents, and only have static ones but the costs of such a move are high. For example, in our case consider the scenario when mobile agent is requested to find some knowledge structures related to the words *arrangement and accounts* from several users computers.

A static single-agent program would need to request for all files residing on the remote knowledge sharing host, which may total to several gigabytes. Each of these actions involves sifting through plenty of extraneous information which could/would clog up the network.

And consider the alternative. JMiningDialog NLU module encapsulates, user sentences to the entire program within an agent which consumes may be only several kilobytes which roams the other hosts included in the knowledge sharing network, arrive safely and queries these hosts locally, and returns ultimately to the home computer. This alternative obviates the high communications costs of shifting, possibly, gigabytes of information to user local computer. Hence, mobile agents provide a number of *practical*, though *non-functional*, advantages, which escape their static counterparts. So their motivation include the following anticipated benefits [1].

1. Reduced communication costs: there may be a lot of raw information that need to be examined to determine their relevance.
2. Limited local resources: the processing power and storage on the local machine may be very limited (only perhaps for processing and storing the results of a search), thereby necessitating the use of mobile agents.
3. Easier coordination: it may be simpler to coordinate a number of remote and independent requests and only collate all the results locally.
4. Asynchronous computing: you can 'set off' your mobile agents and do something else and the results will be back in your mailbox, say, at some later time. They may operate when you are not even connected.
5. A flexible distributed computing architecture: mobile agents provide a unique distributed computing architecture which functions differently from the static set-ups. It provides for an innovative way of doing distributed computation.

We have used aglets mobile agents framework in our implementation. Aglets are Java objects that can move from one host on the network to another and have all features mentioned above. More on this techniques can be found in [18].

5. Conclusion

We presented agent based natural language dialog and understanding architecture for data querying from database management systems and presenting it to the user. We presented reasons why it is important to have in the future, solutions based on mobile agent approach even if now our data amount can be solved by stationary agents approach. Our experience shows that even if we have limited amount of data for teaching process, the right strategies can be found. We believe that integration between agents that extract information from Internet and others unstructured information sources and information delivery software brings optimal solution for companies data analysts. Our research shows that distributed knowledge architecture is more flexible and adaptable for such tasks then centralized solutions.

References

[1] Aglets Specification. *ttp://www.trl.ibm.com/aglets/.*

[2] **I. Androutsopoulos, G.D. Ritchie, P. Thanisch.** Natural Language Interfaces to Databases – An Introduction. *Natural Language Engineering*, 1(1), 1995, 29-81.

[3] **I. Androutsopoulos, G.D. Ritchie, P. Thanisch.** Experience Using TSQL2 in a Natural Language Interface. *J. Clifford and A. Tuzhilin, editors, Recent Advances in Temporal Databases – Proceedings of the International Workshop on Temporal Databases, Zurich, Switzerland, Workshops in Computing, Springer-Verlag, Berlin*, 1995, 113–132.

[4] **P. Atzeni, G. Mecca, P. Merialdo.** Design and Maintenance of Data-Intensive Web Sites, *Proc*. EDBT'98, 1998.

[5] **J.C. Bottraud ,G. Bisson, M.F. Bruandet.** An Adaptive Information Research Personal Assistant. White paper. *http://www.dimi.uniud.it/workshop/ai2ia/cameraready/bottraud.pdf.*

[6] **H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, Y. Wilks.** Experience of using GATE for NLP R/D. *Proceedings of the Workshop on Using Toolsets References* 200 *and Architectures To Build NLP Systems at COLING*-2000, *Luxembourg*, 2000. *http://gate.ac.uk/.*

[7] ELF Software Co. *http://www.elf-software.com*.

[8] **D. Esposito.** Talk to Your Data. White paper. *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnenq/html/mseq75.asp*. 1999.

[9] **A. Fuggetta.** Open source software – an evaluation. *Journal of Systems and Software* 66(1), 2003, 77-90.

[10] **M.N. Huhns, M. Larry, M. Stephens.** Intelligent Agents, in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. *G. Weiss (Ed.), MIT Press, Cambridge, MA*. 1999.

[11] IBM. An Introduction to IBM Natural Language Understanding. *An IBM White Paper,* 2003.

[12] IBM Voice Toolkit V5.1 for WebSphere® Studio. *http://www-306.ibm.com/software/pervasive/voice_toolkit/*, 2004.

[13] Information Builders. Leveraging Your Data Architecture for Enterprise Business Intelligence. *White Paper. http://www.informationbuilders.com*, 2004.

[14] Joone – Java Object Oriented Neural Engine. *http://www.jooneworld.com/*.

[15] **A. Laukaitis, O. Vasilecas, R. Berniunas.** JMining – information delivery web portal architecture and open source implementation. *O. Vasilecas at al (Eds). Information Systems Development. Advances in Theory, Practice and Education. Springer Science*, 2005, 199-206.

[16] **A. Laukaitis, O. Vasilecas.** Mobile agents architecture in data presentation domain. *Edited by A. G. Nilsson et al. (Eds). Advances in information system development, Springer Science*, 2006, 891-902.

[17] **A. Laukaitis, O. Vasilecas, R. Berniunas, E. Augilius.** An architecture for natural language dialog applications in data exploration and presentation domain. *Communications of the Ninth East-European Conference on Advances in Databases and Information Systems ADBIS*, 2005, 135-149.

[18] **A. Laukaitis, O. Vasilecas.** The use of the natural language understanding agents with conceptual models. *Communications of the* 8th *International Conference on Enterprise Information Systems*, 2006, 308-311.

[19] **E. Levin, R. Pieraccini, W. Eckert.** Using Markov Decision Process for Learning Dialogue Strategies. *Proc. ICASSP* 98, *Seattle, WA, May* 1998.

[20] **E. Levin, R. Pieraccini, W. Eckert, G. DiFabbrizio, S. Narayanan.** Spoken language dialogue: From theory to practice. *IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado*, 12-15 December 1999.

[21] **D.J. Litman, M. S. Kearns, M. A. Walker.** Automatic Optimization of Dialogue Management. *White paper*, 1998.

[22] Microsoft corporation. SQL Server and English Query. *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/architec/8_ar_ad_0hyx.asp*, 2003.

[23] Microsoft corporation. Building a Corporate Portal using Microsoft Office XP and Microsoft SharePoint Portal Server. *White Paper*, 2001.

[24] **J. Novak, M. Wurst, M. Fleischmann1, W. Strauss.** Discovering, Visualizing, and Sharing Knowledge through Personalized Learning Knowledge Maps. *White paper*. 2002.

[25] **H. S. Nwana.** The Potential Benefits of Software Agent Technology to BT. *Internal Technical Report, Project NOMADS, Intelligent Systems Research, AA&T, BT Labs, UK* 1996.

[26] Oracle corporation. Oracle9iAS Portal 3.0.9.8.2 Architecture and Scalability. *White Paper*. 2002

[27] **R. Pieraccini, E. Levin, W. Eckert.** AMICA, the AT&T Mixed Initiative Conversational Architecure. *Proc. of EUROSPEECH* 97, *Rhodes, Greece, September* 1997.

[28] **S. Ruwanpura.** SQ-HAL: Natural Language to SQL Translator. *http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura, Monash University*, 2000.

[29] SAS corporation. SAS Information Delivery Portal. *White paper*. 2000.

[30] **N.I. Takeuchi.** The Knowledge-Creating Company. *Oxford University Press*, 1995.

[31] **M. Turunen, J. Hakulinen.** Jaspis – A Framework for Multilingual Adaptive Speech Applications. *Proceedings of 6th International Conference of Spoken Language Processing* (ICSLP 2000), 2000.

[32] VoiceXML Development Guide. *http://www.vxml.org.*

[33] **M. Watson.** Practical Artificial Intelligence Programming in Java. *http://www.markwatson.com*, 2002.

[34] World Wide Web Consortium, Extensible Markup Language. *http://www.w3.org/XML.*

[35] World Wide Web Consortium, Extensible Stylesheet Language. *http://www.w3.org/Style/XSL.*

Received December 2006.

DOI: 10.5755/j01.itc.36.1.11823