A HOTAIR Scalability Model

A. Mur, L. Peng, R. Collier, D. Lillis, F. Toolan, J. Dunnion

Department of Computer Science,
University College Dublin (UCD), Belfield, Dublin 4, Ireland.
{ mur.angel, liu.peng, rem.collier, david.lillis, fergus.toolan,
john.dunnion}@ucd.ie

Abstract. This paper describes a scalable mathematical model for dynamically calculating the number of agents to optimally handle the current load within the Highly Organised Team of Agents for Information Retrieval (HOTAIR) architecture.

1 Introduction

Indexing the World Wide Web is a complex task that requires a software infrastructure that has the ability to crawl through millions of web pages, extracting their content, and storing representations of that content in a form that is most appropriate for retrieval. Traditionally, research in this area has focused upon the development of information retrieval techniques that improve (1) the location and extraction of content, and (2) the representation of that content in forms that engender higher levels of precision and recall.

While this area of research remains one of the key research areas in Computer Science, it is becoming increasingly acknowledged that the design of the architecture in which these techniques are embedded is equally important. In fact, a recent news article on the success of Google made the point that "many people consider the company's operations expertise more valuable than the actual search algorithms that launched the enterprise" [15]. This is reflected in the fact that Google have been able to develop a robust and reliable distributed search architecture [1] that has cost millions of dollars, rather than the tens of millions of dollars that it has cost other competitors.

The design of robust and reliable search engine architectures that can scale effectively over large numbers of machines is a significant engineering problem. This paper presents one approach to solving this problem through the use of intelligent agents [32]. Specifically, it introduces the HOTAIR Search Engine architecture, an *extensible* and *scalable* architecture for the discovery, retrieval and indexing of documents from multiple heterogeneous information sources.

Within the HOTAIR architecture, extensibility is engendered through the design of an architecture that provides support for: (1) the plugging in of multiple indexing strategies such as the Vector Space Model [25] and the Extended Boolean Model [26]; and (2) the ability to rapidly and seamlessly integrate diverse sources of information. This requires the use of an open infrastructure that is able to dynamically

adapt its configuration to seamlessly integrate new techniques and information sources into the system.

Conversely, scalability is engendered through the design of an architecture that can be easily expanded as requirements increase. Typically, this will take the form of increasing the number of machines on which the architecture is deployed. Underlying this is the assumption that adding more machines will deliver an improvement in performance of the system. However, achieving this improvement is often a non-trivial task for a system administrator. It often requires detailed knowledge of both the expected load that will be placed on the system, and the most appropriate configuration for handling that load. Their task is further complicated by the fact that the actual load on such a system will change over time as the number of searches increases and decreases. This can result in the application undergoing significant periods of non-optimal performance. Thus, supporting scalability requires a solution that is flexible, aware of the current level of demand, and which can dynamically adapt its configuration to reflect both changes in demand and the availability of resources.

The key characteristic that the implemented architecture must conform to is the ability to dynamically adapt its configuration as requirements, demand, and resources change. These characteristics are synonymous with the types of system that agent technologies are most suited to. This has led to the development of the HOTAIR Document Indexing System, a multi-agent system that has been designed to adapt its configuration in response to changes in demand, and which supports the seamless integration of new techniques and information sources.

Scalability is an important, yet under-researched, aspect of agent platforms. The dynamics of multi-agent systems are hard to predict and the number of agents in large-scale distributed applications can vary considerably over time.

This paper aims to implement a mathematical model that can be used to estimate the number of agents required based on the available resources.

2 Related Work

The first Internet search engines began to appear in the mid-1990s. One of the first was the World Wide Web Worm (WWWW) [18]. Since their emergence, the main focus of research in this area has been on the development of better information retrieval techniques. Perhaps the most successful of these has been PageRank, the information retrieval technique that underpins the Google search engine [5].

Traditionally, the implementation of search engines, such as Google, was based on cluster-based architectures, with large numbers of low-cost servers located at one or a few locations and connected by high-speed LANs [4]. Their robustness and reliability is commonly achieved through the replication of services across many different machines, and the implementation of an infrastructure that automatically detects and handles failures [1].

Some researchers, such as ODISSEA [29], have explored the potential of Peer-to-Peer technology in the design of a next generation of distributed search architectures. Others have focused on the concept of meta-search [17][27], focusing on the definition of strategies for combining results from numerous search engines.

Another approach to Internet Search is through the use of software agents [13] [14] to perform tasks such as discovering, indexing and filtering documents and routing relevant information to users. By far the most prominent agent-based approach is through the use of single agent systems, which act as assistants that do all the tasks by themselves. For example, POIROT [24] is a web search agent based on relevance, LETIZIA [16] is an agent that assists Web browsing, and CITESEER [2] is an autonomous citation index finding relevant research publications on the WWW.

In contrast, multi-agent systems are decentralized and distribute tasks among a number of agents. ACQUIRE [9], is an example of a mobile agent-based search engine for retrieving data from heterogeneous, distributed data sources. In contrast, AMALTEA [19] is a search tool that discovers and filters information using a multi-agent evolving ecosystem.

Multi-agent systems are highly dynamic. The number of agents can scale up or down to ensure optimal performance [20] [3]. Scalability is also a term that is often used to refer to extensible functionality. SAIRE [21] is a scalable agent-based information retrieval engine because it supports heterogeneous agents.

The problem of scalability and some scaling techniques are described in [31]. An overview of multi - agent system scalability and a labor market application to model scalability can be found in [28].

3 The HOTAIR Indexing System

The HOTAIR Document Indexing System has been implemented using Agent Factory [8], a cohesive framework that delivers structured support for the development and deployment of multi-agent systems, which are comprised of agents that are autonomous, situated, social, intentional, rational, and mobile [7].

A diagrammatical overview of the agents that make up the system architecture is presented in figure 1. The actual number of agents that exist at any time varies depending upon the demand on and the resources available to the system. In addition, these agents are deployed over a number of different agent platforms that reside on different physical machines.

The creation of agents is a service that is provided by the Platform Manager (PM) system agent. Each agent platform contains a PM, which is responsible for handling requests to create more agents. Upon receipt of a request, a PM negotiates with its counterparts to decide on which machine(s) the requested agent(s) should be created. If there are insufficient resources to create all of the requested agent(s), then the PM agents can either refuse or partially fulfil the request.

3.1 The HOTAIR Agents

Within the HOTAIR architecture, the *Data Gatherer* (DG) agents are charged with the task of analyzing information sources. In the current version of the architecture, two types of DG have been implemented: the Collection DGs are used to process documents stored within static Document Collections, while the Web DGs are, in essence, web spiders that are crawling the World Wide Web. All DG agents follow a common behaviour, they search their assigned information source, discovering new

documents and downloading them into a temporary cache. Internally assigned document identifiers are added to an internal queue and the Broker agent is informed of the existence of new documents.

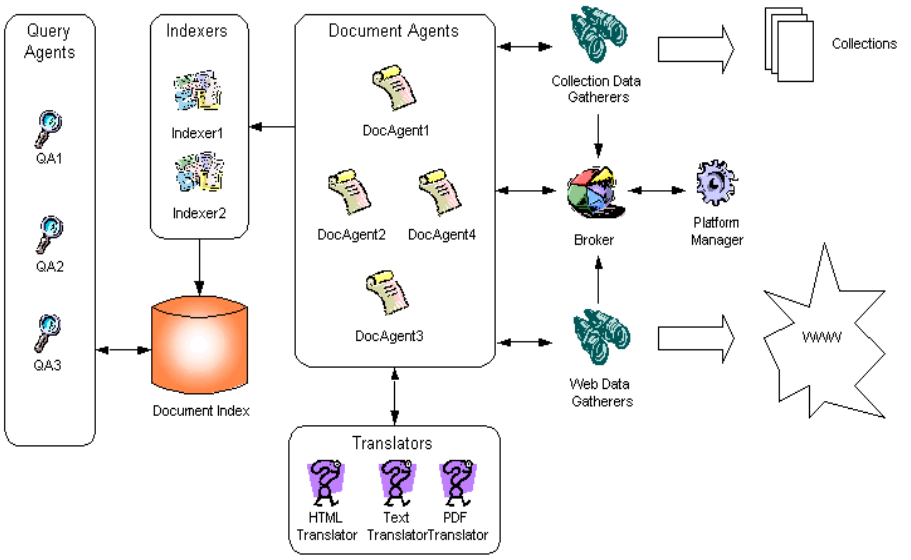

Fig. 1. The HOTAIR Document Indexing System

The *Broker* agent is responsible for monitoring the status of the DG's. This status is currently represented as the size of each DG's document queue. Periodically, the Broker requests a status update from each DG. Whenever a DG's status changes, the Broker reviews how many Document Agents (DAs) to assign to it. If the Broker decides that there are not currently enough DAs, then it asks the local AMS agent to create more DAs. As discussed earlier, this request may be refused. Thus, in cases where the Broker has an insufficient number of DAs, the Broker assigns DAs to DGs that are most in need of additional DAs. When significant disparities exist, the Broker re-assigns some existing DAs to different DGs.

Document Agents (DAs) encapsulate the workflow of the system, that is, they know how to get a document indexed. Currently, indexing a document involves: (1) getting a document from the DG, (2) getting the document translated by a Translator agent, and (3) getting the document indexed by an Indexer agent. Once assigned to a DG, each DA follows the prescribed workflow until either the DG has no more documents or the Broker re-assigns it to another DG. Once an assignment finishes, the Broker either re-assigns the DA or instructs it to terminate itself.

The *Translator* agents are responsible for translating documents from their native format into an internal format, known as the Hotair Document Format (HDF), that is understood by the Indexers. Each Translator specializes in translating one type of document. Currently supported formats include: Portable Document Format (PDF), HTML, Postscript (PS), and plain text. Should demand for a translation service become excessive, a Translator is able to use the Agent Factory cloning capability to clone itself [30]. Excessiveness is currently measured by demand passing a prescribed

threshold. Once created, the load is spread between the original and the clone. The clone is terminated if demand falls below a second lower threshold.

The *Indexer* agent is responsible for indexing documents. Eventually, HOTAIR will support numerous indexing strategies, however, currently it supports only the Vector Space Model. As with the Translators, Indexers are able to clone themselves should demand pass a given threshold.

The final set of agents is the *Query* agents. These agents query the document index on behalf of the user. They provide an agent-oriented interface to the HOTAIR system. In future versions of the architecture, these agents will perform a number of additional activities, including query expansion and user modelling.

4 HOTAIR Scalability Model

The HOTAIR architecture specifies three key points of adaptation: (1) through the cloning of Indexer Agents, (2) through the cloning of Translator Agents, and (3) through the creation of Document Agents.

Document Agent Scalability impacts the speed at which documents are indexed [22]. For a collection of documents, there will be a specific number of Document Agents, for which the document indexing speed is optimal. These agents process this collection more efficiently than other number of agents.

The Broker agent decides the optimal number of Document Agents to process a collection of documents. It uses a formula that represents a Scalable Document Agent Model.

4.1 Scalability model using Multiple Linear Regression

There are two main features of a document collection or group of documents : their size (total number of occurrences i.e. total number of words with repetition) N_o and the number of documents N_d. For each collection, it is possible to explore manually which is the optimal number of agents N_{da} that performs best in terms of time.

The objective of the experiment presented below is to find an equation that allows us to calculate automatically the optimal number of agents from any group of documents.

A solution can be found using a Multiple Linear Regression (MLR) [23]. MLR is a method used to model the linear relationship between a dependent variable and one or more independent variables or predictor variables. MLR is based on least squares: the model is fit such that the sum of squares of differences of observed and predicted is minimized. A general model expresses the value of a variable Y as a linear function of one or more variable X_i and an error term: $Y = b_0 + b_1 X_1 + \ldots + b_k X_k + \varepsilon$ where $\varepsilon \sim N(0,\sigma^2)$. b_0 is a regression constant, b_i is the coefficient on the i^{th} predictor variable X_i, k is the number of variables and ε is the error term.

If we have n experiences, the model $Y_j = b_{0j} + b_{1j} X_{1j} + \ldots + b_{kj} X_{kj} + \varepsilon_j$, (j=1…n), can be compactly written using a matrix notation **Y=XB+E** where

Y= $(Y_1, Y_2, \ldots Y_n)$ **; B=** $(b_1, b_2, \ldots b_k)$ **;**

$$\mathbf{E}=(\varepsilon_1,\varepsilon_2,.....\varepsilon_n)\ ;\ \mathbf{X}=\begin{pmatrix} 1 & X_{11} & . & . & X_{k1} \\ 1 & X_{12} & . & . & X_{k2} \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & X_{1n} & . & . & X_{kn} \end{pmatrix}\ ; \qquad \textbf{(1)}$$

B values can be estimated using the equation $\mathbf{B_s= (X'X)^{-1}X'Y.}$

4.2 Scalable Document Agent Model

We validate the MLR scalability model using Document Agents. Our response would be the optimal number of Document agents N_{da} and our predictor variables N_d and N_o : N_{da} = f (N_d, N_o).The optimal number, N_{da}, represents the number of a group of Document Agents that process documents quicker than other groups for every combination (N_d , N_o). The best group was chosen analysing time processing $t_{(Nd\ ,\ No)}$ of 50 agent groups from 1 agent to 50 agents. Every $t_{(Nd\ ,\ No)}$ was calculated several times and N_{da} is the integer nearest the mean of the results.

The results were obtained using 3 document collections, each of which contained 1000 documents. The first two collections are subsets of the Cranfield and Med collections, while the 3rd collection is comprised of single word documents (i.e. one word per document). This collection is unusual but necessary to get a general model valid for any kind of document.

The table 1 shows a selection from n=30 observations of N_{da} from different combinations (N_d , N_o). The n observations have been selected independently of one another.

N_d	N_o	N_{da}
…….	…….	……
1	26	1
50	227	7
100	6411	15
600	39404	24
1000	65454	32
1000	1000	18
……..	……..	…….

Table 1: Table of the different combination (N_d , $N_{o,}$ N_{da}).

The model found is : $N_{da} = b_0 + b_1 \times N_d + b_2 \times \ln(N_d) + b_3 \times N_d \times N_o$ where b_0 = 1.6563; b_1 = -0.0097; b_2 = 3.5143; b_3 = 2.3364e-007. The Figure 2 shows a 3D representation of the model.

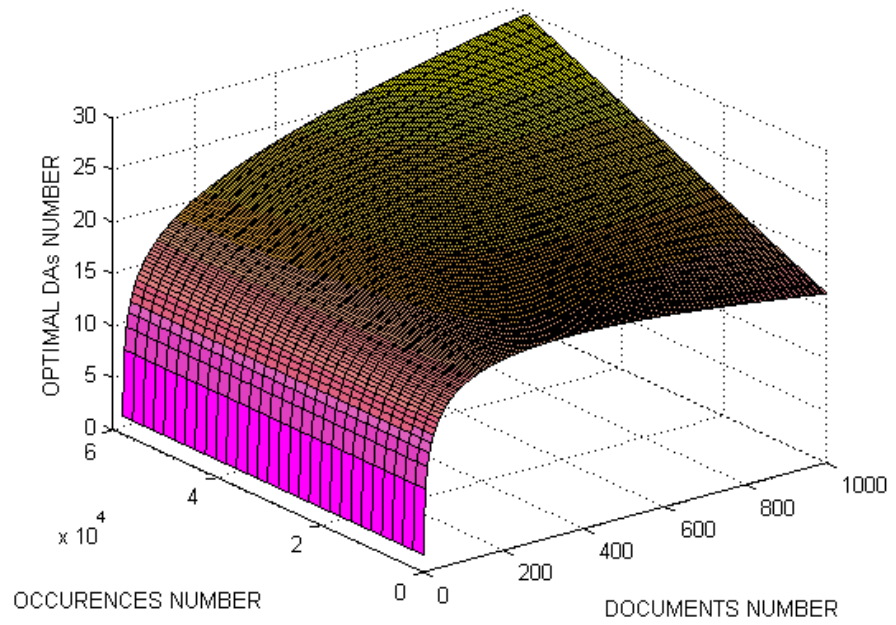

Fig. 2: 3D representation between the Optimal DAs number and the occurrences and documents number.

Table 2 shows the parameters used to validate the model. A Fisher test [23] is used to explain the model utility and a coefficient of correlation R is used to calculate the explanatory power of the regression (2).

R	**F – ratio**	**Significant**
0.9671	F = 110.8083	1%

Table. 2: Validation of the model

$$\mathbf{R} = \frac{\left\| \hat{N}_{da} - \overline{N}_{da} \right\|}{\left\| N_{da} - \overline{N}_{da} \right\|} \; ; \; \mathbf{F} = \frac{(n-k-1)\,\Sigma\left(\hat{N}_{da} - \overline{N}_{da}\right)^2}{(k)\,\Sigma\left(N_{da} - \hat{N}_{da}\right)^2} \; ; \qquad (2)$$

$\hat{N}_{da}$: fitted values, $\overline{N}_{da}$: mean of the $\mathbf{N_{da}}$ observations, N_{da} : observations

The value of R compared with the value 1 suggests that the chosen model has been very successful in relating $\mathbf{N_{da}}$ to the predictors. The Fisher test (F-ratio) shows that we have a significant model and it means there is a useful linear relationship between $\mathbf{N_{da}}$ and at least one of the predictors.

A Student test [23] (t-ratio $T_i = b_i$ / SD(b_i) where SD is the standard deviation of b_i) was used to determinate if all the coefficients of the predictor variables are useful.

Term	b1	b2	b3
Significant	*	**	**

Table. 3: ** means very significant , * significant.

The Student test shows that all our coefficients are useful. Consequently, we have a significant model with the minimum number of predictors.

4.3 Discussion

The model found represents a general view of how the Document Agent community scales as the number of documents to be process changes.

A document for any quantity of occurrences needs one DA always. From a collection of 5 documents the number of agents begins to change in relation to the number of occurrences.

The 3D graph shown in figure 2 shows that the optimal DAs number increases when the documents and occurrences number increases. Document number is more important than the occurrence number. But both are significant from the Student test.

In practice the optimal number of agents is an integer, consequently the estimated float value from the model is rounded to the nearest integer.

This model for documents uses the number of occurrences N_o like a predictor. Due to the high correlation between the N_o and the number of bits of each document N_b, other similar model could be found using N_d and N_b instead of N_d and N_o. This new model should be better for indexing web pages et/or documents dynamically. N_d and N_b can be obtained before processing the documents.

5 Conclusions / Future Work

This paper presents a Document Agent Scalability model for the HOTAIR architecture. This architecture is able to dynamically reconfigure itself to reflect changes in demand through either the creation of additional DAs or through the cloning of Indexer or Translator agents.

The model allows us to study HOTAIR Scalability and automatically gives the optimal number of DAs for any collection of documents. The performance of the HOTAIR architecture improves from a priori knowledge of the optimal number of agents using a model. The Broker agent assigns to the system the optimal number of Document Agents to process a collection of documents.

It is our intention to use the same procedure to build a Scalable Model for other types of HOTAIR agents, namely: Indexer Agents and Data Gatherer Agents.

While these experiments are based on a simplified HOTAIR Document Indexing System, we believe that the results are still valid. In particular, it would seem sensible to assume that, once the optimal number of DAs has been reached for a given indexer, then performance can only be improved by adding another indexer.

References

1. Barroso, L.A, Dean, J., and Holzle, U., *Web Search for a Planet: The Google Cluster Architecture*, in IEEE Micro, 23(2):22-28, 2003.
2. Bollacker k. ; Lawarence, S; Giles C.L.: *Citeseer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications*, Agents'98, 2nd International ACM Conference on Autonomous Agents, 116. 1998.
3. Brazier, F., Van Steen, W.: *On MAS Scalability*. Second International Workshop on Infrastructure for Agents, MAS and Scalable MAS. 2001.
4. Brewer, E.: *Lessons from giant scale services*, IEEE Internet Computing, pages 46-55, August, 2001.
5. Brin, S., and Page, L., The Anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Networks and ISDN Systems, 30(1-7): 107-117, 1998.
6. Chen, L; Sycara, K.: *WebMate : Personal Agent for Browsing and Searching*, Proceedings of the Second International Conference on Autonomous Agents, St. Paul, MN, May, 132-139. ACM Press, New York, NY, 1998.
7. Collier, R., Agent Factory: A Framework for the Engineering of Agent-Oriented Applications, PhD Thesis, Dept. Computer Science, University College Dublin, 2001.
8. Collier, R., O'Hare, G. M. P. Lowen, T. D., and Rooney, C. F. B., *Beyond Prototyping in the Factory of Agents*, In Proc. 3rd Int. Central and Eastern European Conference on Multi-Agent Systems (CEEMAS), Prague, Czech Republic, 2003.
9. Das, S., Shuster, K., and Wu, C.: *Agent-Based Complex Query and Information Retrieval Engine*. AAMAS'02. July 15-9,Bologna, Italy, 2002.
10. Doorenbos, R. B., Etsioni, and Weld, D.S.: *A Scalable Comparison-Shopping Agent for the WWW*, in W.L. Johnson and B. Hayes –Roth (eds). Proc, Proceedings of the First International Conference on Autonomous Agents pp. 39-48, Marina del Rey, CA, USA. ACM Press, 1997.
11. Fischmaister, S., Vigna, G., and Kemmerer, R.A.: *Evaluating the Security of Three Java-based Mobile Agent Systems*, Proceedings of the Fifth International Conference on Mobile Agents, Springer, pp,31-41, 2001.
12. FIPA, The FIPA 2000 Specifications, FIPA Website URL: http://www.fipa.org, Accessed May 2005.
13. Julian, V; Rebollo, M; Carrascosa, C.: *Agentes de Informacion*. Revista Base 37. ISSN 1135-0695, 56-62., 2001
14. Klusch, M.: *Information agent technology for the Internet: a survey*, Data and Knowledge Engineering, volume 36 (3), 337-372. 2001.
15. LaMonica, M., *Google's Secret of Success? Dealing With Failure*, CNET News.com,URL:http://news.com.com/Googles+secret+of+success+Dealing+with+failure/2100-1032_3-5596811.html, 2005.
16. Lieberman H. *Letizia: An Agent That Assists Web Browsing*, Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, 1995.
17. Mamma, The Mamma Meta Search Engine, URL: http://www.mamma.com, 1996
18. McBryan O. A'*GENVL and WWWW: Tools for Taming the Web.* First International Conference on the World Wide Web, CERN, Geneva (Switzerland), May 25-26-27 1994.

19. Moukas A., Maes P., *Amalthea: An Evolving Multiagent Information Filtering and Discovery System for the WWW*, invited paper for the first issue of the Journal of Autonomous Agents and Multiagents 1998
20. Neuman, B.: *Scale in Distributed Systems*. In T.Casavant and Singhal (eds). Readings in Distributed Computing Systems, pp 463-489. IEEE Computer Society Press, Los Alamitos, CA., 1994.
21. Odubiyi, J.B., Kocur, D. J., Weinstein S. M., Wakim, N., Srivastava, S., Gokey, C., and Graham, J.: *SAIRE- a scalable agent-based information retrieval engine*, in Proceedings of the first international conference on Autonomous agents, pp. 292-299, Marina del Rey, CA USA, feb. 1997.
22. Peng L., R. Collier, A. Mur, D. Lillis, F. Toolan, J. Dunnion. Self-Configuring Agent-based Document Indexing System, CEEMAS 2005, Budapest, Hungary, 15-17 September, 2005.
23. Peck R, Devore J. Statistics, the exploration and analyses of data, Duxbury Press. 1997.
24. Ramirez, J., Donadeu, J., and Neves, F., *Poirot a relevance-based web search agent*, AAAI Workshop Artificial Intelligence for Web Search. 2001.
25. Salton, G. and Lesk, M.E.: *Computer evaluation of indexing and text processing*. Journal of the ACM, 15(1):8-36, January 1968.
26. Salton, G., Fox, E. A., and Wu, H.. *Extended Boolean information retrieval*. Communications of the ACM, 26(11):1022-1036, 1983.
27. Smyth, B., Freyne, J., Coyle, M., Briggs, P., Balfe, E. (2003) I-SPY - Anonymous, Community-Based Personalization by Collaborative Meta-Search, Proceedings of the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI-2003). Cambridge, 2003.
28. Song, R., and Korba, L., *The Scalability of A Multi-agent System in Security Services*. NCR/ERB-1098, NRC 44952,August, 2002,.
29. Suel, T., Mathur, C., Wu, J., Zhang, J., Delis, A., Kharrazi, M., Long, X., Shanmugasundaram, K., *A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval*, in Proc. 12th International World Wide Web Conference, Budapest, Hungary, 2003.
30. Tong,Y., O' Hare, G. M. P., and Collier, R., *Using agent uml to design cloning in agent factory*, in Proceedings of the 1st Workshop on COnceptual MOdelling for Agents (COMOA 2004), Shanghai, China, 2004.
31. Wijngaards, N.J.E., Brazier, F.M.T., van Steen, M., Distributed Shared Agent Representations. Multi – Agent –Systems and Applications II Vol: 2322, pp.213-220, July, 2002, Springer Verlag, Lecture Notes in Computer Science.
32. Wooldridge, M., and Jennings, N. R., *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review 10(2), 1995.