Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2007 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2007

Flexibility of Multiagent Problem-Solving Based on Mutual Understanding

Stefan Kirn
Universität Hohenheim

Vijayan Sugumaran
Oakland University

Flexibility of Multiagent Problem Solving based on Mutual Understanding

Stefan Kirn
Information Systems II
Universität Hohenheim
Stuttgart, 70593, Germany
kirn@uni-hohenheim.de

Vijayan Sugumaran
Department of Decision and Information Sciences
School of Business Administration
Oakland University
Rochester, MI 48309
sugumara@oakland.edu

Abstract

Multiagent technology provides concepts, architectures and cooperation protocols for applications with challenging demands for the flexibility of the overall system behavior. A recently developed flexibility framework identifies six dimensions of multiagent system (MAS) flexibility: qualitative flexibility, quantitative flexibility, problem solving flexibility, economic flexibility, time flexibility and configuration flexibility. These flexibility concepts support the systematic description, analysis, and explanation of MAS flexibility, and the design of flexibility profiles into MAS applications. This paper introduces this flexibility framework and discusses how to incorporate problem solving flexibility into an agent-based information retrieval system. Based on the prototype implementation, a "mutual understanding layer" is proposed as part of the MAS architecture. An outline of future research completes the presentation.

Keywords

Agent-based information retrieval, flexibility framework, flexibility architecture, multiagent systems, multiagent technology, semantic cooperation

Introduction

In the last few years we have not only experienced a tremendous explosion in the complexity of technical systems, enterprise structures, supply chains, and customer demands, but also market challenges, balance (and its dynamics) of power between the different stakeholders of companies, environmental and political systems (Dietrich et al. 2007). These trends require us to implement the optimization, adaptation (learning), and the overall behavior of socio-economic systems in a much more decentralized and networked manner, which will encompass more bottom-line oriented approaches than we usually employ today. Besides the necessary cultural changes in our economies (and societies), this will also require new construction principles, architectures, and behaviors for our information-technology based systems.

Multiagent technology is one of the enabling technologies that is supposed to provide appropriate concepts, architectures and protocols for applications with challenging flexibility demands. Based upon the results of 20 years of research, a recently developed flexibility framework (Kirn 2006) has identified and formalized six dimensions of MAS flexibility: qualitative flexibility, quantitative flexibility, problem solving flexibility, economic flexibility, time flexibility and configuration flexibility. These flexibility concepts support the systematic description, analysis, and explanation of MAS flexibility, and they support the design of flexibility profiles into multiagent system-based applications.

This paper addresses the flexibility of multiagent systems in the information retrieval domain. It has increasingly become difficult for users to retrieve relevant web pages from queries. To address this problem, the Semantic Web has been proposed to extend the WWW by giving information well defined meaning. The Semantic Web relies heavily on ontologies to provide taxonomies of domain specific terms and inference rules for a body of knowledge that serves as a surrogate for semantics (Berners-Lee et al. 2001). Intelligent agents are being used in Semantic Web applications for a variety of tasks including being able to interpret relevant semantics on web pages (Hendler 2001). However, many of the agent based applications have experienced limited success because of the inflexibility of these systems and the agents not being able to adapt to different situations. Hence, there is a great need for designing flexible agent-based systems that can respond to changes in the problem solving environment.

The overall aim of this research is to identify the relevant flexibility demands and provide suggestions for the design and control of cooperatively executed information retrieval tasks in a typical web application. The general perspective of the paper is design-oriented, i.e., it develops a flexibility-based MAS architecture for web querying. The specific objective of this research is to develop a flexible architecture for a MAS that implements a heuristics based methodology for intelligent retrieval of information from the web by taking into account semantics in users' requests and available ontologies. The contribution of doing so is an architecture that supports the requirements of problem solving flexibility within a MAS and provide a system that will help realize the full potential of the semantic web by taking a pragmatic approach to processing user's queries.

The remainder of the paper is organized as follows. The next section provides a brief background on multiagent systems and flexibility of MAS. It is followed by the formalization of agents and MAS based upon their problem solving behavior. The succeeding section discusses the flexibility framework, which includes the six dimensions of flexibility. The next section discusses the example application of information retrieval on the Web and provides an architecture for a MAS that incorporates some of the flexibility dimensions. A heuristics based approach for information retrieval and the prototype implementation is also discussed. The subsequent section provides a brief discussion on the impact of the MAS mutual understanding layer on the flexibility dimensions. The final section concludes the paper and outlines some future work.

Literature Review

Multiagent Systems

A multi-agent system (MAS) is defined as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver (Durfee and Lesser 1989). The increasing interest in MAS research is due to significant advantages inherent in such systems, including their ability to solve problems that may be too large for a centralized single agent, provide enhanced speed and reliability and tolerate uncertain data and knowledge. Some of the key research issues related to problem-solving activities of agents in a MAS are in the areas of coordination, negotiation and communication (Nwana 1996). Coordination is the process by which an agent allocates tasks to other agents and synthesizes the results from these agents to generate an overall output. Negotiation is the process by which agents solve their conflicts and reach a compromise. For coordination and negotiation, they need to communicate with one another and hence the system should provide a general communication mechanism.

In order to support MAS requirements such as coordination, negotiation, and communication, several architectures have been reported in the literature. The Gaia methodology (Wooldridge et al. 2000; Zambonelli et al. 2003) supports the development of agents with formal models of knowledge and environment, role hierarchies, and representation of social structures and relationships. However, it requires that inter-agent relationships and agent abilities to be static at run time and hence may be of less value in the unpredictable of domain of Internet applications. RETSINA (Reusable Task Structure Based Intelligent Network Agents) (Sycara et al. 1996; Sycara et al. 2003) is a general-purpose modeling framework which proposes goal, role, context and attitude as first class objects for modeling multi-agent systems in an open world. The Multiagent Systems Engineering (MaSE) methodology leads the designer from the initial system specification to the implemented agent system (DeLoach et al. 2001). In the analysis phase, MaSE uses goal hierarchies and role models, whereas the design phase creates agent-class, communication and deployment diagrams. AUML (Bauer et al. 2001) is an extension to UML to represent various aspects of agents by introducing new types of diagrams including agent class diagrams and protocol diagrams. Although these methodologies support cooperating agents, they do not support teams of agents very well.

The Tropos methodology (Bresciani et al. 2004; Giunchiglia et al. 2002) has a strong focus on requirements analysis and consists of different phases, namely, early and late requirements, architectural design, detailed design and implementation. However, one criticism of this approach is that it does not provide strong support for protocols and modeling the dynamic aspects of the system (Dam and Winikoff 2003). The Prometheus approach (Padgham and Winikoff 2002) supports software engineering activities and detailed processes for developing agent applications. It consists of the following three phases: (a) system specification, (b) architectural design, and (c) detailed design. While this method is targeted for people who do not have a background in agents, it is weak in terms of support for concurrency. Finally, DECAF (Distributed, Environment-Centered Agent Framework) (Graham et al. 2003) is an agent toolkit to design, develop and execute agents to achieve solutions in complex software systems. Since DECAF creates solutions to high-level tasks via decomposition using a predefined library of task schemas, it has the shortcoming of not being able to create new task decompositions.

The methodologies and architectures discussed above primarily take an implementation point of view and focus heavily on developing a system rapidly. However, due to their emphasis on the implementation viewpoint, these architectures fall short in non-functional capability considerations of systems development. For example, the implementation of a facilitator or a coordinator agent within a system may lead to bottleneck problems if sufficient attention is not paid to internal process control structures. In MAS development, not only functional aspects but also quality attributes, such as reliability, adaptability, etc. should be considered as main issues.

Flexibility and MAS

Flexibility is one of the most important attributes of multiagent systems (Rosenschein 1985; Bond and Gasser 1988). Multiagent system flexibility emerges from the interplay of the different components of agents (body of knowledge, local problem solving capabilities, interaction capabilities, etc.) with the different components of the overall system (i.e., common language, openness, cooperation and coordination protocols, effectiveness of cooperation topologies, scalability, etc.), and with its environment. It depends further on the richness of dynamic interactions between local activities and global activities (micro-macro-link) (Conte and Castelfranchi 1995).

A particular degree of MAS flexibility can emerge from very different sets of flexibility components (design alternatives). Due to its inherent complexity, we still lack not only a validated theory but also a generally accepted definition of the flexibility of multiagent systems (Kirn et al. 2006). In order to approach this important problem for MAS design, Kirn (2006) has suggested a six-dimensional flexibility framework for multiagent systems (see the flexibility framework section). This paper makes use of this framework in order to design and study the flexibility profile of a typical multiagent system application, i.e. the cooperative retrieval of information on the World Wide Web based upon mutual understanding of the involved agents.

The following section discusses how agents behave and provides a formalization of agent behavior within a multiagent system.

Formalization of Agents and MAS

We define the concept of agents based upon their problem solving behavior. Agents perform their actions in a three-step cycle, as given below:

1. Information reception: agents observe (*sense*) their environments via sensors to identify the relevant information as *sensory_input* $\in$ *Sensory Inputs* and transfer them into one or more *perception* $\in$ *Perceptions*.

$$sense: SensoryInputs \rightarrow Perceptions \quad (1)$$

2. Information processing: in the second step, the agent draws conclusions and updates its *internal_state* $\in$ *InternalStates*. The state $internal\_state_{new}$ is normally dependent on the state $internal\_state_{old}$ and a specific set of *perceptions*:

$$reason: InternalStates \times Perceptions \rightarrow InternalStates \quad (2)$$

   The definition of *InternalStates* is based upon the environment model *EnvModel*, the *Goals*, and the *Commitments* (already promised services to be fulfilled in future) of the agent:

$$reason: (EnvModel_{old} \times Goals_{old} \times Commitments_{old}) \times Perceptions \rightarrow (EnvModel_{new} \times Goals_{new} \times Commitments_{new}) \quad (3)$$

3. Actions: In the third step, the agent fulfils its commitments by performing *action* $\in$ *Actions* in its environment:

$$act: Commitments \rightarrow Actions \quad (4)$$

The behavior of an agent can thus be described by sequences of the mappings *sense, reason* and *act*. By definition, each agent is a member of the overall agent community. The agent community is given by:

$$AgentCommunity := (A \times C) \quad (5)$$

where $A := \{a_1, ..., a_i, ..., a_n\}$ is the set of agents existing in the network and $C := \{c_{ij} \mid c_{ij} \rightarrow (a_i, a_j); 1 \leq i, j \leq n\}$ is the set of directed communication channels between pairs of agents. Please note that $c_{ij} \neq c_{ji}$. C defines the overall connectivity of the agent society. Necessarily, interaction requires communication. Thus, c_{ij} also represents the set of possible interactions of the action(s) of agent a_i with the sensor(s) of agent a_j.

Multiagent systems are subsets of *AgentCommunity*. They are defined as

$$MAS_i \subseteq A \times C \quad (6)$$

In accordance with the literature, we assume that MAS_k is established only if and only if a cooperative problem solving process is required (dynamic definition). MAS_k declines, iff problem solving is completed. We can therefore assume that if an agent accepts the invitation to join a cooperative problem solving process, it is immediately bound to this problem solving process. The binding of an agent can be static (static planning) or dynamic. If an agent decides to provide a specific service s_i to this problem solving process, this service s_i is bound to this process, too. This establishes a second level of binding.

The overall system behavior of a multiagent system is thus manifested in six steps:

Step I: Selecting agents $a_i \in A$ that will be invited to join cooperative problem solving.
Step II: Binding agents $a_i \in A$ to constitute MAS_k.
Step III: Selecting services $s_i \in SetOfServices$ needed for problem solving.
Step IV: Binding services $s_i \in SetOfServices$ to the process of problem solving.
Step V: Performing the problem solving process.
Step VI: Decline of the MAS_k.

This is a generic definition of a multiagent system, supporting our analysis of the flexibility framework introduced in the next section. To this end, it abstracts from many conceptual and technical details (e.g., agent platform, common agent language, ontologies, coordination and cooperation protocols, directory services, etc.) needed to develop a concrete multiagent system.

Flexibility Framework

Building on the formal models of agency and multiagent systems above, Kirn (2006) has suggested a formalized MAS flexibility framework. This framework consists of six flexibility dimensions:

- *qualitative flexibility:* which services can be provided?
- *quantitative flexibility:* how often can a service be provided, e.g., in a given time interval?
- *problem solving flexibility:* what different ways of reasoning (problem solving strategies) can be used to solve a problem at hand?
- *economic flexibility:* how much or which resources are required to perform the required changes?
- *time flexibility:* how fast can required adaptations of a MAS can be put in place?
- *configuration flexibility:* how easily, and to what degree, can the current structure of a MAS be transformed into another one?

In the following section we formalize the MAS flexibility framework.

Qualitative Flexibility

Qualitative flexibility *QualFlex* denotes that set of services s_i, $1 \leq i \leq n$ (service portfolio) which a multiagent system can deliver to its environment. A quantitative measure of qualitative flexibility is the size of this set:

$$QualFlex: SetOfServices \rightarrow |N \quad (7)$$
$$qual\_flex := |SetOfServices| \quad (8)$$

This definition still ignores that the elements of *SetOfServices* may be different in many aspects. They may be unequal with respect to their relevance, they may be requested in different frequency, at different times, in different quantities, etc. Depending on the requirements of a particular application it may thus be necessary to employ more sophisticated flexibility measures, e.g., such as statistical distribution functions.

In general, not all elements of *SetOfServices* are known. The set *SetOfServices* may even change over time. This may cause all well-known problems of incomplete, vague and non-monotonic knowledge. Similar considerations hold for all flexibility definitions below.

Quantitative Flexibility

Quantitative flexibility *QuantFlex* denotes how often a multiagent system can provide each of its services s_j per unit of time. Assumed, agent a_i can provide a set of services s_j $(1 \leq j \leq n)$ k times per unit of time. Then, a simple definition of quantitative flexibility is given by:

$$QuantFlex: SetOfServices_i \rightarrow SetOfServices_i \times |N \quad (9)$$

$$quant\_flex(s_1, \ldots, s_j, \ldots, s_n) := \{(s_j, k_j) \mid j := 1, \ldots, n, k \in |N\} \quad (10)$$

This definition ignores that the elements of *SetOfServices* may be different with respect to their relevance for a given demand of the environment and that the quantitative output of service s_j may change over time Additional formalisms may thus be needed to get more appropriate specifications of *QuantFlex*.

Example: Assumed, agent a_i provides three services s_1, s_2 and s_3. Each service can be provided up to k_j times. This leads to $QuantFlex := \{(s_1, k_1), (s_2, k_2), (s_3, k_3)\}$. However, the triples *(100, 100, 100)* and *(1, 1, 10.000)* for k_j, $j := 1, \ldots, 3$ describe very different quantitative flexibilities that cannot easily be compared.

Problem Solving Flexibility

A multiagent system i may be able to produce a service s_i in different ways, i.e. through different sequences of activities. We call these sequences the problem solving patterns of a MAS i with respect to a particular service s_j, denoted as $Patterns_{i,j}$. We can thus define:

$$ProblSolvFlex_i: Patterns_{i,j} \rightarrow |N \quad (11)$$

$$probl\_solv\_flex_i(s_j) := |Patterns_{i,j}| \quad (12)$$

Again, this is still a simple definition. It ignores, for example, that there may be different pre- and post-conditions for each pattern, that they may require different quantities and qualities of resources, etc. This, however, depends on the underlying application and the concrete specification of the multiagent system under consideration.

If one aims to use this concept, a new reasoning component must be integrated into the architecture of a multiagent system. The inference process works as follows:

1. *Self observation*: The multiagent system has to keep track of all steps performed during problem solving.
2. *Evaluation*: The MAS evaluates these after the end of a problem solving process.
3. *Learning*: Store learning results as a new pattern in the problem solving pattern database.
4. *Monitoring*: As multiagent systems typically exist in dynamic environments, they are supposed to continuously monitor the applicability of the pattern in their databases.
5. *Use*: If necessary, employ an already existing pattern to solve a current task.

Economic Flexibility

We have already stated that flexibility cannot be taken for granted. Three main types of costs, all together denoted as potential expenses *pe,* can be identified:

1. *Resources R*: Flexibility may require additional resources (computing time, storage, bandwidth, access to external knowledge, etc.). These resources can be limited, they can cost money, they can be controlled by someone else, etc. If a multiagent system aims to involve such resources it may have to pay a price for it.
2. *Risks for current commitments CC*: As a result of changed internal structures and/or behaviors, the risk may arise that current commitments are broken (e.g., through a crash of ongoing problem solving processes, decreased performance, bottleneck problems, limited scalability, delays of service deliveries, reduced service quality, etc.). In such cases, these risks $Risk_{CC}$ have to be calculated in appropriate economic terms, e.g., in monetary equivalents.
3. *Changes of external behaviors E*: Flexibility may lead to changes in behaviors of the environment. This may require additional internal activities and may even complicate the internal planning and reasoning processes. This also can be (at least, in principal) mapped to appropriate (monetary) terms.

We call these costs the economic flexibility of a MAS. If *curr* denotes the current state of a multiagent system, *adapted* the state after adaptation, and M the additional costs (e.g., in terms of monetary equivalents), then a first definition of EconFlex can be given by:

$$M := R \times CC \times E \tag{13}$$

$$EconFlex: SetOfServices_{curr} \times M \rightarrow SetOfServices_{adapted} \tag{14}$$

$$econ\_flex(s_i) := \{ s_j \mid s_j := s_i \circ (r_x, cc_y, e_z) \wedge (r_x, cc_y, e_z) \in M \wedge x = 1, \ldots, m, y = 1, \ldots, n, z = 1, \ldots, o)\} \tag{15}$$

Please note that different combinations of additional resources $(r_x, cc_y, e_z) \in M$ may transform a service s_i to the same new service s_j. This case can be considered as a combination of economic flexibility and problem solving flexibility.

Time Flexibility

Time flexibility denotes the amount of time necessary to perform required adaptations. This is of particular interest if a multiagent system needs to perform adaptations in very short time. May $t \in |N$ denote the number of units of time (e.g., seconds) needed to perform a requested adaptation of service s_i. Then, a simple definition of time flexibility is given by:

$$TimeFlex: SetOfServices_{curr} \times |N \rightarrow Boolean \tag{16}$$

This definition does only denote, that the MAS under consideration is able to perform a particular adaptation within time span *t*. This can be expressed through:

$$time\_flex(s_i, t) := \begin{cases} TRUE & iff\ "adaptation\ successful" \\ FALSE & otherwise \end{cases} \tag{17}$$

In the long term, multiagent systems must be able to perform necessary adaptations faster than changes occur to their environments.

Configuration Flexibility

According to the definitions in the previous section, a multiagent community *AgentCommunity* is given by:

- $A := \{a_1, a_2, \ldots, a_n\}$: set of agents, each able to sense, reason and act
- $C := \{c_{ij} \mid c_{ij} \rightarrow (a_i, a_j);\ 1 \leq i, j \leq n\}$: set of directed communication channels between pairs of agents a_i, a_j
- *AgentCommunity* := (A × C)
- Multiagent systems MAS_i are defined as subsets of *AgentCommunity*: $MAS_i \subseteq A \times C$

Configuration flexibility is a mapping from the current structure of a MAS to a new structure. For our model of multiagent systems, this is denoted by the following definition:

$$ConfigFlex: A_{curr} \times CC_{curr} \rightarrow A_{adapted} \times CC_{adapted} \tag{18}$$

Let $\underline{A}$ be the set of agents which has been removed together with the set of agents which have been added and let $\underline{C}$ be the set of directed communication channels which has been removed together with the set of communication channels which have been added. Then we write:

$$config\_flex(MAS_i) := A \times C \tag{19}$$

If and only if $config\_flex(MAS_i) = \varnothing$, then MAS_i is called static flexible.

We have provided a generic definition of multiagent systems. This abstracts from many conceptual and technical details (e.g., agent platform, common agent language, ontologies, coordination and cooperation protocols, directory services, etc.) needed to develop a concrete multiagent system. In a concrete case, thus, the above definition must be extended to the conceptual model of the multiagent system(s) under development.

Semantic Cooperation Example

In this section we discuss a multiagent system that incorporates the flexibility concepts discussed in the previous section. This multiagent system deals with the information retrieval from the web. The continued growth of the World Wide Web makes the retrieval of relevant information for a user's query increasingly difficult. Current search engines provide the user with many web pages, but have varying levels of relevancy. In response, the semantic web has been proposed to retrieve and use more semantic information from the web. In this research, an intelligent semantic web retrieval architecture is presented to automate the processing of a user's query while taking into account the query's context. The multiagent architecture uses a heuristic based methodology that combines techniques to support meaning based searches including natural language processing, ontology integration, semantic network processing, and agent based reasoning.

Since the Web is a highly heterogeneous environment, the MAS architecture used for information retrieval from the Web should be highly flexible so that it can make use of available resources. The natural language query expressed by the user has to be parsed, disambiguated, and expanded with semantic and conceptual information in order to improve the precision of the documents retrieved. For query parsing, the system can make use of natural language parsing services available on the Web. Similarly, for disambiguating query terms, various ontologies and lexicons available on the Web can be used and the query can be refined with appropriate terms. Contextual and semantic information can also be added from user profiles other knowledge sources. Thus, the design of a MAS for Web querying should take into account the flexibility concepts discussed in the previous section in order to facilitate effective use of available resources for information retrieval. The agents that comprise this architecture should work cooperatively and be customizable depending upon the task at hand. This also requires that the agents have a certain level of mutual understanding. This is even more critical if the agents are acquired from different sources and integrated into the system.

The proposed multiagent system architecture is shown in Figure 1. It utilizes existing research on natural language processing, WordNet, and ontologies to extend the Semantic Web, and builds upon a proven search engine technology (Google). The overall aim is to automate the search process to improve users' efficiency and effectiveness in obtaining online knowledge. The architecture consists of six components: user, interface agent, inference agent, knowledge base, online resources agent, and query constructor agent. The interface agent has three roles: 1) capturing a users' natural language query for processing, 2) interfacing with other components, and 3) presenting the final results to the user. Ultimately, the system should require only limited interaction with the user. In order for the agent architecture to scale up and work in different scenarios and domains, there should be considerable semantic cooperation between the agents. At the very least, the architecture should have qualitative flexibility, problem solving flexibility, economic flexibility and configuration flexibility.

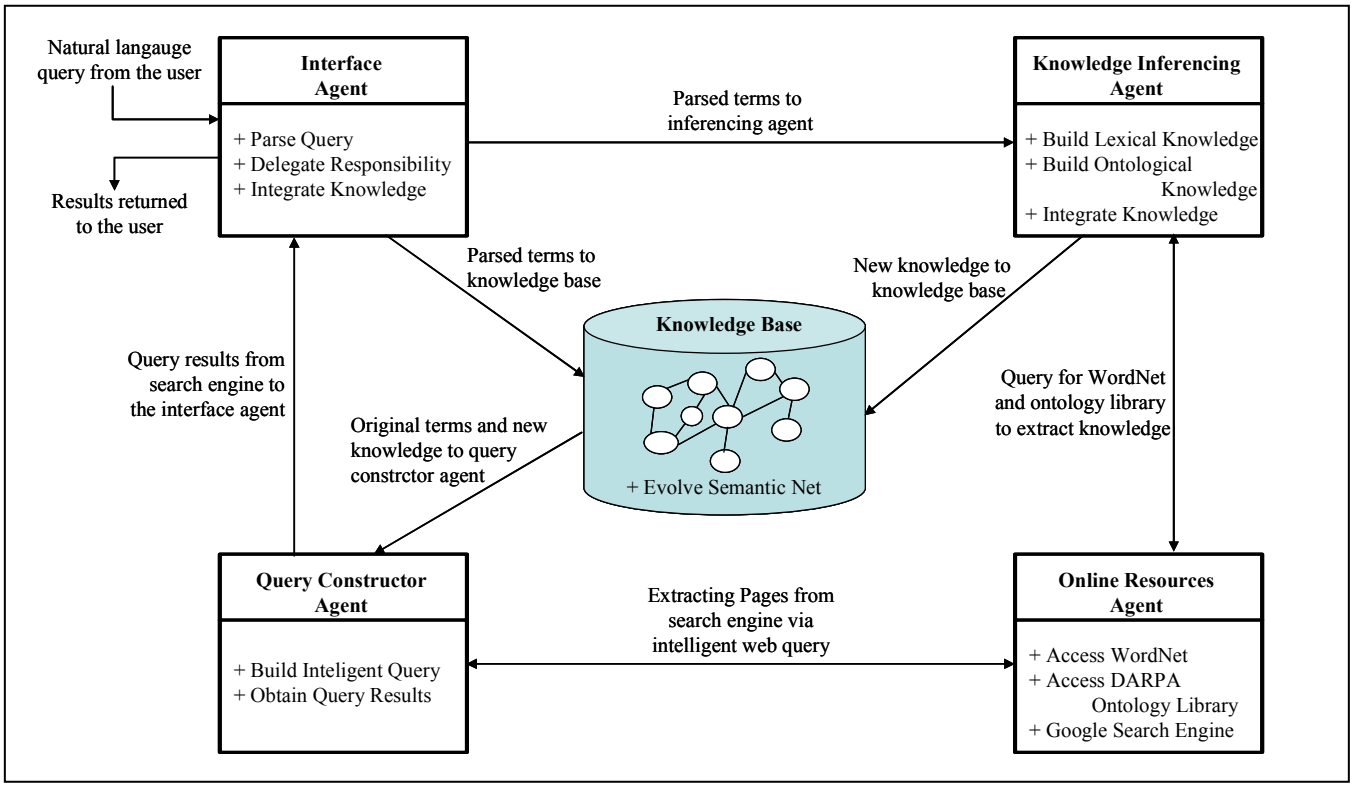

Figure 1. Intelligent Semantic Web Retrieval Architecture

Table 1: Semantic Query Heuristics

Component	Objective	Heuristics
Interface Agent	To transform query for processing	• Heuristic 1 – Proposition parsing (separate natural language into propositions via grammatical markers and AND/OR components) • Heuristic 2 – Stop word and verb removal
Knowledge base	To build semantic network of terms in query	• Heuristic 3 – Candidate links (create links based upon sequencing of terms in query)
Inference engine	To expand knowledge of query terms using WordNet and the ontology library and infer semantic links between terms in the knowledge base	• Heuristic 4 – Phrases (consecutive terms queried as pairs to determine if word-pair has meaning) • Heuristic 5 – Redundancy (terms that are a subset of another or have nearly identical lettering are ignored) • Heuristic 6 – Knowledge access (to access knowledge on each query term from WordNet and ontologies) • Heuristic 7 – Match terms in knowledge base (links are formed between terms with matching ontological or lexical information) • Heuristic 8 – Inconsistency (if two links are logically inconsistent, the first link is assumed to be correct)
Query constructor	To construct and execute a semantic query from the knowledge base	• Heuristic 9 – Boolean (determine whether to use OR in query) • Heuristic 10 – Phrase (terms found as a phrase in WordNet or ontology library queried as phrase) • Heuristic 11 – Negative knowledge (web search narrowed by excluding incorrect word senses) • Heuristic 12 – Iteration (iterate the query until a low number of pages can be returned that are maximally relevant)

The knowledge base utilized in the architecture is represented using a semantic network. This semantic network grows through the introduction of knowledge from the inference agent and shrinks by pruning excess knowledge. The query constructor agent processes the original query terms and new knowledge from the knowledge base is used to construct intelligent web queries. It uses a set of heuristics and an interface tool to construct web queries in the syntax required by Google. Using knowledge sources to disambiguate and extend query expressions is not a new topic. Information retrieval (IR) and library science researchers have studied the issues for some time (Voorhees 1994). Our architecture aims to apply this philosophy using newly developed knowledge sources and search engines to provide a pragmatic software solution for WWW users (Chang and Hsu 1998). Our approach is novel in the sense that the architecture supports the different dimensions of flexibility and uses a heuristics based solution approach. Table 1 shows the heuristics that form the basis of the intelligent query processing methodology.

The heuristics in Table 1 represent three types: (a) natural language processing (heuristics 1 and 2), (b) semantic net evolution (heuristics 3 through 8), and (c) query construction (heuristics 9 through 12). It is important to note the constraints imposed on these heuristics by our pragmatic approach. Typical IR studies test the effectiveness of various search algorithms on a known corpus (collection) of documents (Raghavan 1997). Our aim is to test the benefit of IR and intelligent systems techniques in practice for web users. We thus selected Google as our search engine given that it currently crawls more web pages and is most used in practice. This introduces two differences between typical IR research and ours: (1) our queries are not tested across a *known* corpus of documents as Google currently indexes over 2 billion web pages; and (2) Google imposes constraints on the nature of the queries it accepts (i.e., it offers limited syntax and will only query up to ten terms). The heuristics in Table 1 are designed to create the most effective web query given these constraints. Our approach is to verify the benefit of the heuristics as an overall set, and the relative benefit of each one. We also want to assess whether a fully- or semi-automated execution of the heuristics is optimal. A prototype has been developed to illustrate the feasibility and benefits of the semi-automated approach (Voorhees 1994).

Implementation

A prototype has been developed using JADE (http://jade.tilab.com/) and J2EE technologies (javabeans, servlets and jsp). The prototype follows the traditional three-tier client-server architecture as shown in Figure 2. The client is a basic web browser, through which the user specifies search queries in natural language. The server contains agents and Java application

code while the database backend contains the WordNet files. The prototype also uses webservices such as Teemapoint's NL Parser (www.teemapoint.com) and Google search engine (www.google.com).

The prototype consists of three agents: a) Input-Output-Parser Agent, b) WordNet Agent, and c) Query Refinement and Execution Agent. These were implemented using a combination of jsp pages and servlets. The input-output-parser agent is responsible for capturing user input, parsing the natural language query, and returning results. The agent performs URL encoding of the query, sends it to Teemapoint's parser servlet, and requests an XML document that contains the parsed text. The returned XML document is validated and parsed after which noun phrases are extracted. Processing of the XML document is performed through DOM (Document Object Model) classes, interfaces, and methods included in JAXP (Java API for XML Processing). Based on the noun phrases (propositions) identified, an initial search query is created.

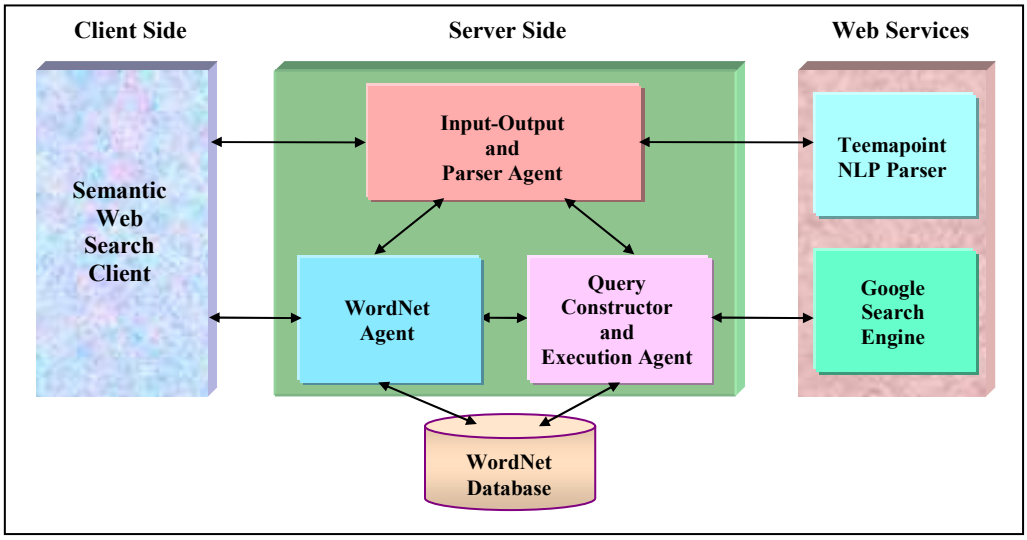

Figure 2. Initial Prototype Design

The WordNet Agent interfaces with the WordNet lexical database via JWordNet (a pure Java standalone object-oriented interface available at http://sourceforge.net/projects/jwn/). The prototype uses WordNet 1.6 (PC). For each noun phrase, the agent queries the database for different word senses and lets the user select the most appropriate sense for the query. The agent extracts word senses, synonyms and hypernyms (superclasses) from the lexical database and forwards them to the query refinement agent to augment the initial query.

The Query Refinement and Execution (QRE) agent expands the initial query based on word senses, and synonyms obtained from WordNet. The refined query is then submitted to the Google search engine and results returned to the user. The agent interacts with Google through its Web API service. If Google receives more than ten terms in a query, terms after the tenth are ignored. The API also restricts the user to a maximum of 1000 pages, in blocks of ten pages at a time. The QRE agent applies heuristics from Table 1 to augment the initial query. For example, it searches for phrases (word pairs) and includes them within double quotes, adds synonyms (from the WordNet synset) to the query based on the word sense selected by the user, and adds negative knowledge (terms) from the remaining word senses. It also keeps track of the number of terms currently in a search query and executes appropriate heuristics to expand the query. The refined query is then sent to Google. The next version will include Jess (Java Expert System Shell) to provide additional reasoning capability (http://herzberg.ca.sandia.gov/jess).

Discussion

Mutual Understanding-Enhanced MAS Flexibility Architecture

This paper has presented a six-dimensional framework of MAS flexibility. A basic assumption of this framework is that all agents in a MAS are able to understand each other. This however is not the case in reality, where agents are designed and implemented individually, existing in different domains as well as in different social and organizational contexts. In the implementation of our web querying system, one of the difficulties we faced was the fact that the parsing services were changing over time and the available ontologies and lexicons were also evolving. Hence, our agents and the overall architecture needed to be flexible in order to accommodate these variations. Based on our experience in implementing the web querying system, we suggest that in order to enhance the capabilities of the MAS framework, an additional semantic layer is needed to

support mutual understanding of the different agents currently in the system and the agents that may join the MAS at runtime. The resulting flexibility architecture for a MAS is depicted in Figure 3. As shown in Figure 3, the agent community may consist of several multiagent systems or a federation of loosely coupled heterogeneous agents. If multiple agents from this community are imported into a particular MAS, in order for these agents to cooperatively work together, the MAS should have a "MAS Mutual Understanding Layer" which greatly facilitates the interoperability of these agents. This additional layer will, to a large extent, impact the various flexibility characteristics of the MAS architecture.

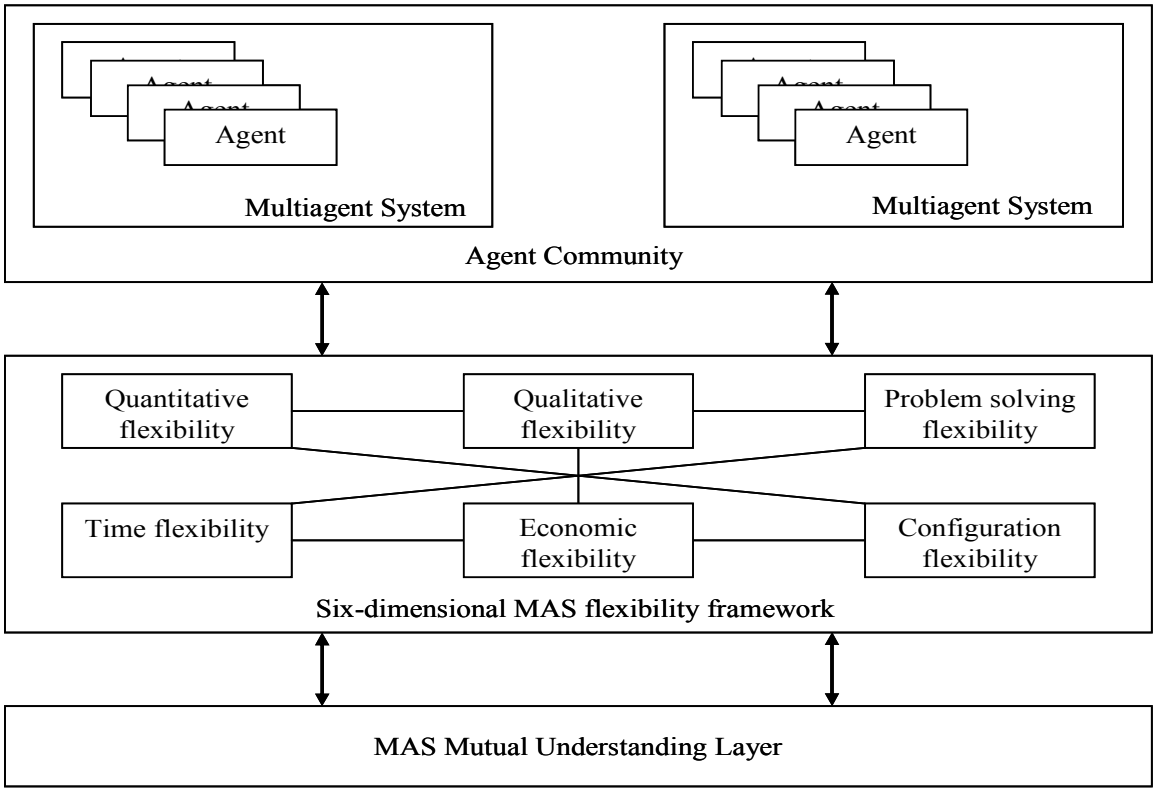

Figure 3. Mutual Understanding-enhanced Flexible Architecture for MAS

Considering the six flexibility dimensions discussed earlier, we briefly mention how the mutual understanding layer might contribute to MAS flexibility. We contend that the mutual understanding layer will impact the various flexibility dimensions in the following manner. However, these hypotheses have to be empirically validated.

Qualitative flexibility: Positive impact - the mutual understanding layer will have a positive impact on this dimension. There is a significant potential to increase the qualitative flexibility of a MAS by mutual understanding as it supports the flexible integration of agents into an existing multiagent system independently of their individual backgrounds, terminologies, etc.

Quantitative flexibility: No direct impact - the quantitative flexibility is not directly affected by the mutual understanding layer (but: see "configuration flexibility" below).

Problem solving flexibility: No direct impact - the problem solving flexibility is also not directly affected (but: see "configuration flexibility" below).

Economic flexibility: Negative impact - the economic flexibility is affected in general as it typically requires resources to cope with different terminologies.

Time flexibility: Negative impact - the time flexibility will be affected in general as it typically requires additional time to cope with different terminologies.

Configuration flexibility: Strong positive impact - There is an enormous potential to increase the configuration flexibility of a MAS by mutual understanding as it strongly supports the flexible integration of agents independently of their individual backgrounds, terminologies, etc. into an existing multiagent system. Further, this may lead to indirect benefits for quantitative and problem solving flexibility.

Conclusion and Future Research

Flexibility is an important characteristic for a multiagent system. In order to better understand this phenomenon, this paper has discussed a MAS flexibility framework. This framework essentially consists of the following six dimensions: qualitative, quantitative, problem solving, economic, time, and configuration. We have formalized each of these dimensions and these flexibility concepts have been utilized in designing a multiagent system for web querying. The architecture of this system contains an interface agent, knowledge inferencing agent, online resources agent, and query constructor agent. These agents cooperatively work together in order to search and retrieve appropriate documents (web pages) from the web in response to a user query. These agents use external resources in accomplishing their tasks. Potentially, they could interact with agents from other information retrieval systems, particularly in the context of cross-lingual and multi-lingual web querying.

In a multiagent system, the autonomy of agents enables them to plan, perform and evaluate their actions according to their internal states, goals and resources. They are aware of the state and dynamics of their external environments and are capable of individually exhibiting situated behaviors. They coordinate their actions with other agents belonging to the same multi-agent system. New agents can join an existing multiagent system and members of a multiagent system may even withdraw at runtime. However, one thorny issue is the fact that if an agent wants to join a coalition, it needs to understand the other agents in that coalition and vice versa. Based on our experience in implementing a scalable web querying multiagent system, we suggest that a MAS Mutual Understanding Layer be incorporated into the MAS architecture to improve interoperability. This layer provides the translation and mapping between the various agents and their capabilities. Our future work involves an in depth empirical investigation of the impact of the mutual understanding layer on each of the flexibility dimensions and implementing reasonably complex multiagent systems in different domains in order to generalize the conclusions drawn. Our future work also includes designing and implementing an efficient MAS mutual understanding layer for different scenarios and application domains. Finally, the framework will be used to evaluate existing multiagent systems with respect to their flexibility characteristics.

References

Bauer, B., Muller, J. P., and Odell, J. "Agent UML: A formalism for specifying multiagent software systems," International Journal of Software Engineering and Knowledge Engineering, (11:3), 2001, pp. 207–230.

Berners-Lee, T., Hendler, J., and Lassila, O. "The Semantic Web," Scientific American, May 2001, pp. 1-19.

Bond, A.; Gasser, L. "An Analysis of Problems and Research in DAI," In Bond, A.; Gasser, L. (eds.), Readings in Distributed Artificial Intelligence, Morgan Kaufman Publishers, San Mateo CA, 1988, pp. 3-35.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. "Tropos: An agent-oriented software development methodology," Autonomous Agents and Multi-Agent Systems, (8:3), 2004, pp. 203–236.

Chang, C.H., and Hsu, C.C. "Hypertext Information Retrieval for Short Queries," Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop, Taiwan, Nov. 1998.

Conte, R., and Castelfranchi, C. Cognitive and Social Action. UCL Press, London, United Kingdom, 1995.

Dam, K. H., and Winikoff, M. "Comparing agent-oriented methodologies," 5th International Bi-conference Workshop on Agent-Oriented Information Systems (AOIS'03), July 2003, Melbourne, Australia , pp. 79–94.

DeLoach, S. A., Wood, M. F., and Sparkman, C. H. "Multiagent systems engineering," International Journal of Software Engineering and Knowledge Engineering, (11:3), 2001, pp. 231–258.

Dietrich, A. J., Kirn, S., and Sugumaran, V. "A Service-oriented Architecture for Mass Customization – A Shoe Industry Case Study," IEEE Transactions on Engineering Management, (54:1), 2007, pp. 190-204.

Durfee, E. H., and Lesser, V. "Negotiating task decomposition and allocation using partial global planning," In L. Gasser, and M. Huhns (eds), Distributed artificial intelligence (Volume II), Pitman Publishing/Morgan Kaufmann, London/San Mateo, CA: 1989, pp. 229–244.

Giunchiglia, F., Mylopoulos, J., and Perini, A. "The tropos software development methodology: Processes, models and diagrams," Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, July 15–19, 2002, Bologna, Italy .

Graham, J., Decker, K., and Mersic, M. "DECAF—A flexible multiagent system architecture," Autonomous Agents and Multi-Agent Systems, (7:1), 2003, pp. 7–27.

Hendler, J., "Agents and the Semantic Web," IEEE Intelligent Systems, 2001, pp. 30-36.

Kirn, S. "Flexibility of Multiagent Systems," In Kirn, St., Herzog, O., Lockemann, P., and Spaniol, O. (eds.), Engineering of Multiagent Systems – Intelligent Applications and Flexible Solutions, Springer, Berlin, 2006, pp. 53-69.

Kirn, St., Herzog, O., Lockemann, P., and Spaniol, O. (Eds.), Multiagent Engineering – Theory and Applications in Enterprises, Springer, Berlin, 2006.

Nwana, H. S., "Software Agents: An Overview," Knowledge Engineering Review, (11:3), 1996, pp.1-40.

Padgham, L., and Winikoff, M. "Prometheus: A methodology for developing intelligent agents," Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, July 15–19, 2002, Bologna, Italy .

Raghavan, P. "Information retrieval algorithms: a survey," Proceedings of the 8th annual ACM-SIAM symposium on Discrete algorithms, New Orleans, Louisiana, 1997, pp. 11 - 18.

Rosenschein, J. Rational Interaction: Cooperation Among Intelligent Agents. PhD thesis, Computer Science Department, Stanford University, Stanford, California, March 1985.

Sycara, K., Decker, K., Pannu, A., Williamson, M., and Zeng, D. "Distributed intelligent agents," IEEE Expert-Intelligent Systems and Their Applications, (11:6), 1996, pp. 36–45.

Sycara, K., Paolucci, M., Van Velsen, M., and Giampapa, J. (2003). The RETSINA MAS infrastructure. Autonomous Agents and Multi-Agent Systems, (7:1), 2003, pp. 29–48.

Voorhees, E. "Query expansion using lexical-semantic relations," Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 1994, pp. 61-69.

Wooldridge, M. J., Jennings, N. R., and Kinny, D. "The Gaia methodology for agent-oriented analysis and design," Autonomous Agents and Multi-Agent Systems, (3:3), 2000, pp. 285–312.

Zambonelli, F., Jennings, N. R., and Wooldridge, M. "Developing multiagent systems: The Gaia methodology," ACM Transactions on Software Engineering and Methodology (TOSEM), (12:3), 2003, pp. 317–370.