

An Application of Knowledge Management using Intelligent Agents

K. Greer¹, D. Bell², H. Wang¹, Y. Bi² and G. Guo¹

1. The School of Computing and Mathematics, The University of Ulster at Jordanstown, email: krc.greer,h.wang,g.guo@ulster.ac.uk
2. The School of Computer Science, The Queen's University of Belfast, email:da.bell,y.bi@qub.ac.uk

Abstract

In this paper we shall outline an approach for using some new technologies to create a knowledge management system that uses both explicit and tacit knowledge to provide a service for a user. The technologies used are Communities of Practice, Web Services and agents. Explicit knowledge will be provided from a Content Repository or from the internet through Web Services and then tacit knowledge held in the minds of the experts shall be provided through Communities of Practice. Intelligent agents shall be used as the software paradigm to implement the system. A user will firstly inspect freely available explicit knowledge held in a Content Repository or retrieved from the internet. He will formulate some preconditions for a service based on this knowledge. The experts will then be queried and asked to apply their tacit knowledge which cannot be codified. Based on their response the service will or will not be performed. This system is to be developed as part of the much larger ICONS knowledge management system as an Intelligent Agent Development Environment (IADE) module.

Correspondence address: Kieran Greer, The University of Ulster at Jordanstown,
Shore Road, Newtownabbey, Belfast BT37 0QB, Northern Ireland.

Telephone: +44 (0) 28 90368209

Email: krc.greer@ulster.ac.uk

1. Introduction

Knowledge can be described as being either explicit or tacit [7]. Explicit knowledge is structured and can be codified. It is the kind of knowledge that can be represented in a form that can be stored in a computer, in a knowledge base for example. Tacit knowledge exists only in people's minds. It has a personal quality that makes it hard to articulate and cannot be codified. In an organisation both types of knowledge may exist. We could have explicit knowledge held in databases or knowledge bases that would be freely available to anyone and tacit knowledge held only in the minds of the experts. They would have to be personally queried to be able to use this knowledge. This paper will describe one part of a larger system that will use both explicit and tacit knowledge to provide a service for a user. The explicit knowledge will be held in a Content Repository, or on the internet which can be queried using Web Services [17]. The tacit knowledge will be held in Communities of Practice [19]. The user will both query the Web Services and use the Communities of Practice through intelligent agents [12]. Some features of agents are useful for the system. There will be some degree of autonomy and intelligence coded into the agents. Also, the communication abilities of agents will be used, as will their tendency to be long running software entities. This system is to be developed as part of the ICONS (Intelligent CONtent management System) project [9]. ICONS is an advanced Knowledge Management System. At the heart of ICONS is an XML based Content Repository which can be accessed through an advanced user interface. The Content Repository can also be queried through a deductive database (DLV [3]) turning it into a knowledge base. For our part of the system we are interested in providing the user with a service. This service is provided by an expert, but the user needs to firstly construct preconditions under which the service should take place. The user could query the Content

Repository or retrieve documents or other information through Web Services to get a general idea about the service and the preconditions under which it should be fulfilled. This explicit knowledge would be freely available, but may not be as up to date as an expert's knowledge or may be missing vital elements that could not be codified. After formulating the preconditions the user would enter into a Communities of Practice process to ask an expert or group of experts to carry out the service. This would involve a number of experts satisfying the preconditions set by the user and the user then selecting the best service offered among one or more options. In our system this selection process will be automatic and will be carried out by a knowledge-based system. In ICONS the Communities of Practice are to be applied to the Structural Fund projects [18] provided by the European Community, which provide an opportunity for poorer states to close the social and economic gap between these countries and the European Community. As stated in [18], even a 1% improvement in the structural fund project proposal acceptance level for Poland would mean an additional 138 million Euro to be invested in the Polish regional economy alone in the years 2004-2006. In this paper we will firstly look at the technologies involved and describe the research areas. Then we can consider the selection process in more detail and how it fits into the ICONS system. Finally we will consider some related work and the stage of implementation.

2. The Agent Platform

As Jennings states [12], 'Agent-based computing represents an exciting new synthesis both for Artificial Intelligence (AI) and, more generally, Computer Science. It has the potential to significantly improve the theory and practice of modelling, designing, and implementing computer systems'. Agents should be intelligent, having some

knowledge of the environment they are working in and be autonomous or semi-autonomous, being able to react to changes in their environment and initiate actions on their own. Another feature of agents is that they use a communication language to communicate with each other, and in doing so form complex relationships with each other. Also they can be long running, which helps them to be autonomous and carry out actions by themselves. These features will be used to varying degrees in the agents suggested in the following sections. Agents run in an agent platform. For ICONS we will be using the JADE agent platform [10].

The JADE agent platform complies with FIPA standards [5]. The platform includes some standard components coded as agents and a message transport system. Figure 1 describes the standard agent management model, where this diagram can also be found in the JADE *programmersguide* document.

The following components are included in the agent platform:

- Agent Management System (AMS) – exerts supervisory control over access to and use of the Agent Platform. This provides a white page and life-cycle service, maintaining a directory of agent identifiers (AIDs) and agent states.
- Directory Facilitator – this provides the default yellow page service in the platform. It is used to look up agents.
- Agent Communication Channel (ACC) – or Message Transport System controls the exchange of messages within the platform or between platforms.
- Agents – the user defined agents created by extending the JADE Agent class.

A feature of agents is their ability to communicate with each other. To do this they use an Agent Communication Language. The FIPA Agent Communication Language (ACL) is based on speech act theory: messages are actions, or *communicative acts*, as they are intended to perform some action by virtue of being sent. Speech act theory is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions. Agents can be created that communicate using ACL and their actions/interactions can be defined by Behaviours or Protocols. We will be using simple behaviours in our agents as well as the FIPA defined Contract Net protocol.

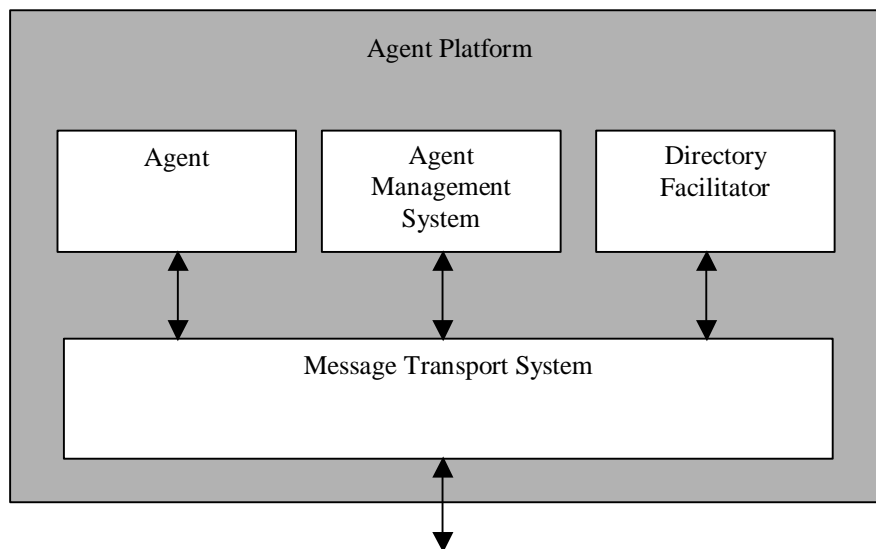


Figure 1. JADE agent platform

3. Communities of Practice

Communities of Practice are described in [8], [13], [18] and [19]. The Communities of Practice group human experts into networks where they have similar goals and a shared understanding of their activities. All Communities of Practice depend on communication and organisations can support this by providing these communication capabilities as well as making it easy for prospective members to find and join them. According to Wenger, McDermott and Snyder [19], Communities of Practice are groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis. Over time they develop a unique perspective on their topic as well as a common body of knowledge, practices and approaches. It may not be possible to codify the knowledge they develop and often it is this knowledge that is the most important. These informal networks serve many purposes, including resolving the conflicting goals of the institution to which they belong or solving problems in more efficient ways. The deep understanding of knowledge developed in Communities of Practice can be difficult for competitors to replicate. In a Structural Fund project the Communities of Practice might relate to risk analysis of infrastructure investment projects.

As specified in [18], one of the applications of the IADE will be to implement a Communities of Practice, and to do this we will have a number of User and Expert agents which will negotiate using the FIPA [5] standard Contract Net protocol. A User will contact a particular User agent passing it the preconditions that will be used as the Call for Proposals for the Contract Net protocol. The User agent will then pass the preconditions to all of the listed Expert agents. These would then decide whether

they could meet the preconditions. This would be done automatically by the agents and so intelligence and knowledge of the expert's domain would need to be coded into the agents. The expert agent will relate whether or not it meets the preconditions by sending or not sending a proposal to the user agent in an appropriate ACL message. The User agent would then evaluate each proposal it receives and decide which one was the best. To the best Expert agent or agents it would send an accept message and reject messages to those not selected. The user agent would then wait for the reply from the accepted agent(s). Upon receiving an accept message, the accepted Expert agent would enter into a manual process. The expert or group of experts represented by the Expert agent would negotiate with each other through email to decide on a contract to meet the user's needs. The Expert agent would be paused waiting for this process to complete. Once the experts had formulated a contract they would send it to the Expert agent, which would return it to the User agent. If the experts failed to agree on a contract they could send a failure message to the User agent. On receiving a contract the user could inspect the contract and if he agrees, could ask the expert(s) to carry out the service. This would terminate the agent activities.

Figure 2 gives a description of the architecture for using a Communities of Practice. From this diagram we can see that the IADE is passed a process activity (one of the proposed forms of passing information) in an XPDL (XML Process Definition Language [20]) file. The IADE then creates an in-process agent, which is passed the XPDL file. An in-process agent is an agent that can be launched from your program and can then be used to communicate with agents running permanently in the system. The in-process agent creates the preconditions from the XPDL file, connects to the

main container(s) where the permanent agents are running and passes to a User agent the preconditions. The User agent contacts Expert agents that are specified and starts to negotiate with them concerning the service. The Expert agents may have contact with experts that are external to the system.

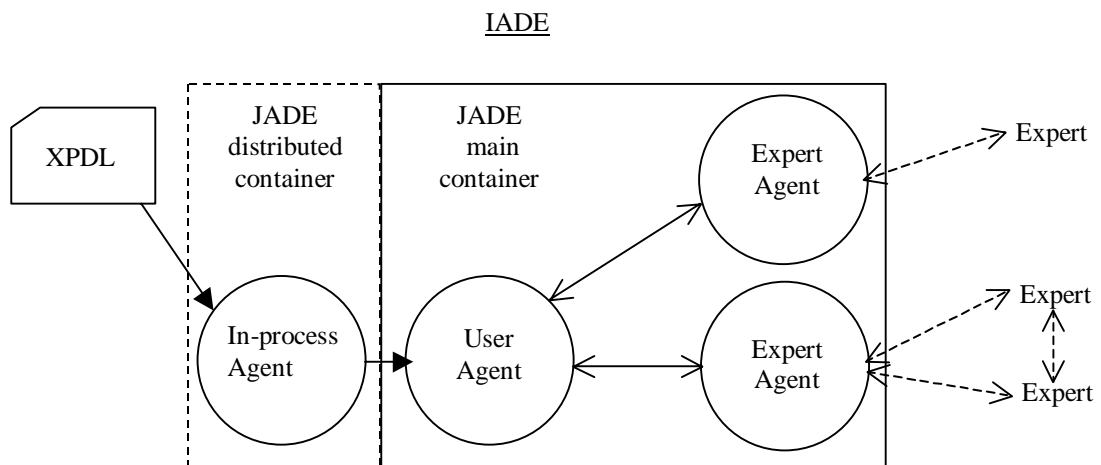


Figure 2. Diagram showing how JADE is used to initiate a Communities of Practice process.

4. Web Services

4.1 Technologies Used

A Web Service is an application or information resource that can be accessed using standard Web protocols [17]. A number of XML-based technologies are used to interface with Web Services and the registries that they register with. The advantage of using Web Services is that they provide a uniform and programmable interface to services over the internet and are becoming the standard. Although the number of Web Services available at the moment is limited we expect this number to rapidly grow and provide services in many different domains. A Web service registers its

services with a UDDI (Universal Description Discovery and Integration) registry. The transport protocol for connecting to the Web Service is called SOAP (Simple Object Access Protocol) and the format for describing the services a Web Service provides and how to connect to them is called WSDL (Web Services Description Language). The specification for SOAP 1.1 can be found at <http://www.w3.org/TR/SOAP/> and for WSDL at <http://www.w3.org/TR/wsdl>. When a service consumer wants to use a service, he queries the UDDI registry to find a service that matches his query, and obtains a WSDL description of the service and also the access point of the service. The service consumer then uses the WSDL description to construct a SOAP message with which to communicate with the service. To connect to a Web Service we will be using the Java Web Services Development Pack, downloadable from the Java web site [11]. In the first instance, we will only be concerned with retrieving information from Web Services. Creating your own Web Service and publishing information on it is a whole new domain. For retrieving information from Web Services and registries we can use the following XML-based technologies: JAXR (Java API for XML Registries), JAXM (Java API for XML Messaging) and SAAJ (SOAP with Attachments API for Java). JAXR is an XML-based technology that provides a standard way to access business registries and discover Web Services. Any user of a JAXR client may perform queries on a registry. There are two public UDDI registries provided by Microsoft and IBM that can be used to search for Web Services. JAXM sends SOAP messages over the Internet in a standard way and allows the sending of non-XML attachments. SAAJ is the basic package used by JAXM for SOAP messaging, which contains the API for creating and populating a SOAP message.

4.2 Searching for Web Services

We will provide an agent to connect to the UDDI registries to look up possible Web Services to be used. The information retrieved from the registries will be stored in a database in a number of tables. The structure of these tables is given in figure 3 below.

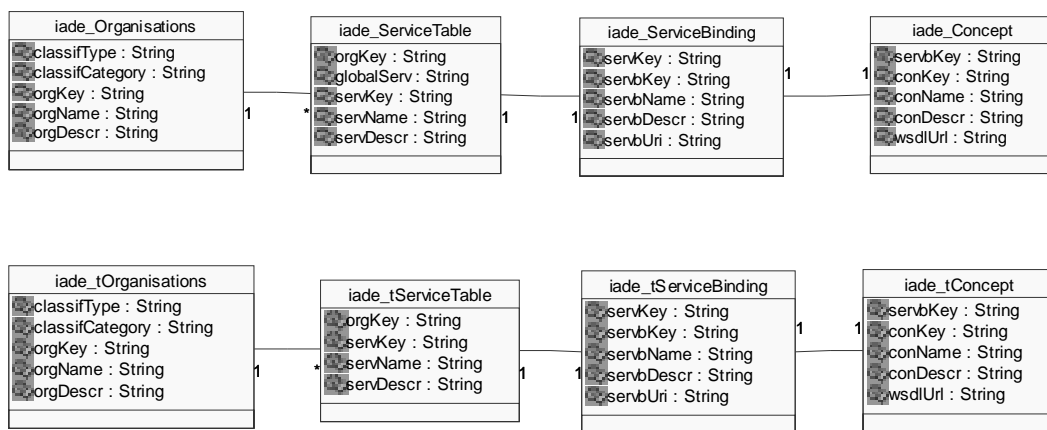


Figure 3. Database structure showing permanent and temporary tables to store the search results. The temporary tables are shown second with the 't' prefix

There are four permanent and four temporary tables. The first table type stores information about an organisation. The second stores information about the services the organisation provides and the third stores information about a binding of the service, that points to technical information about the service. In our case the service binding gives the entry point to access the service over the internet. Finally the concept table may define a tModel (defines an abstract service) that may point to a WSDL document that describes an abstract service type. The user of the system would use a search agent to search for Web Services based on search criteria passed in the form of an XML file. The results of the search would then be stored in the

temporary tables. The user would then use a GUI to inspect the information stored in the temporary tables and look up the Web Services in question. If it is discovered that they provide the required service the information would be transferred to permanent tables from where the Web Services agents could retrieve it.

4.3 Calling Web Services

It is certainly possible to create an agent that connects to a particular Web Service. More challenging is to create an agent that connects to a particular type of Web service or any Web Service. In this case the calls to the Web Service would be heterogeneous and would need to be created dynamically. There are some technologies to do this. With the JAX-RPC (Java API for XML Remote Procedure Call) dynamic invocation interface (DII), a client can call a remote procedure even if the signature of the remote procedure or the name of the service are unknown until runtime. Because of its flexibility, a DII client can be used in a service broker that dynamically discovers services, configures the remote calls, and executes the calls. It is not clear however exactly how dynamic DII can be. Another technology that allows dynamically calling Web Services is the Apache Web Services Invocation Framework [23]. WSIF enables developers to interact with abstract representations of Web services through their WSDL descriptions. WSIF allows stubless or completely dynamic invocation of a Web service, based upon examination of the meta-data about the service at runtime. However these technologies seem to suffer from the same problem. To call a Web Service you firstly have to construct the Java classes to represent the data type(s) of the calling method. This means that the Java object must be created at some time before the call and then used by the program in the call and this seems like a two stage process. In fact, as the WSIF documentation states: ‘The

DynamicInvoker doesn't support invocation of services using complex types since this requires that java representations of the complex types be generated first'. We suggest trying a different and simpler approach. We will try to dynamically construct a SOAP message from a WSDL document and then use SAAJ to send this message to the Web Service. We do not need to create any new classes to represent the complex data types but we need to dynamically construct the structure of the data types in the SOAP message. Also, there are some restrictions like creating the header for the SOAP message. However if the Web Services follow a standard format most will use the same header information. The data returned from the call may also be complex, but this can either be converted into XML or in some cases into a Java object. To allow an automatic process requires some manual input from the user before the Web Service can be called. We need to construct ontologies to define the concepts that define the Web Service calls. For example, if we have an operation called 'average' in one WSDL document and 'mean' in another we need to tell the agent that these two things are the same so that it can successfully construct the Web Service call. The idea of tModels is supposed to address this issue and make services more homogeneous. A service type, defined by a construct called a tModel, defines an abstract service. Multiple businesses can offer the same service type, all supporting the same service interface, that implements the tModel. At the moment Web Services do not seem to be so homogeneous, and so we suggest some mechanism of inputting ontologies to the agent (different from the JADE ontology) for different classifications representing the same concepts. A GUI has been written to help with creating and storing ontologies in the system.

McIlraith et. al. in [15] have developed a system using agents for automatic Web Service discovery, execution and composition. They note some of the problems listed here, but use the semantic web markup language DAML-S to construct a knowledge base to define the Web Services concepts. We will try a simpler approach to create ontologies to define the Web Service concepts. The ontology information will be stored in the relational database tables described in figure 4 below.

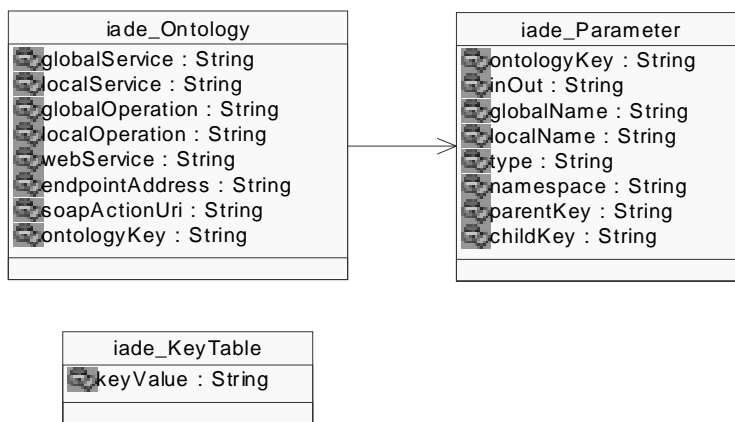


Figure 4: Relational tables used to store ontology details

This ontology really only stores the information required to make a Web Service call. The information stored in the ontology table is the service names, the operation names, the Web Service name and Web Service URLs required for connection. The global names are used by the global ontology and are what the agent understands, while the local names are what each Web Service understands. The Parameter table stores the structure of the parameters required to make the operation call. An ontology key relates this table to the ontology table. We also record if the parameter is an input or output parameter, its name (global name is used to link to the global ontology), type and namespace. There are then two key values relating parent parameters with child parameters, which allows for any structure to be recorded. If the data type is an

array its value will start with ArrayOf as it is done with the WSDL specification. The rest of the specification will be the data type, for example, an array of floats will be ArrayOfFloat. The child key of one concept would be the same as the parent key of another concept if the second concept is a sub-element of the first. So if given a global service and operation and Web service name, we could retrieve only the relevant parameters. This mechanism should allow for the definition of both content and structure in the ontologies. The current implementation allows for automatic parsing of WSDL documents to construct the values in these tables.

5. Using the Agents for Knowledge Management

5.1 Using a Search Agent

To use the explicit and tacit knowledge the agents would be used in a certain order. There is also a construction process required to enter data into the system in a form that can be used by the agents. Before we can use the Web Services agents we need to decide which Web Services we are going to use. The first stage in this process is to look up possible Web Services using the search agent. The user enters a profile describing the type of Web Service he is looking for and then contacts a search agent to allow it to look up UDDI registries to find Web Services that match the criteria. Possible Web Services are found and a certain amount of filtering can be done to extract useless information. The UDDI registries contain information relating to services that are not Web Services so we can check that a URI for the service entry point exists and a URL for the WSDL document exists that ends in 'wsdl' (the most common end to a URL indicating a WSDL document). This will filter out a lot of useless information (for example, the registries contain test entries that do not relate to

real Web Services) and a number of specific search parameters are provided to allow the search to be narrowed further. The profile of the type of Web Service required could be saved and the search agent would periodically look up the registries for new Web Services that had been registered. Also it could check for any Web Services that had de-registered. The autonomous and long running nature of agents would be useful in this process. After a search, the user would manually look up the Web Services through the service entry point and inspect the actual services that they offer.

5.2 Constructing an Ontology

If a Web Service is found that is useful then the ontology for the Web Service needs to be constructed so that it can be used. This is also a manual process but a lot of it is automatic. The advantage of creating an ontology is that it means that the same agent can call more than one Web Service and we do not need to write a new agent to do this. If similar Web Services use the same tModel then this process would not be required but this does not seem to be the case at the moment. So the ontology structure needs to be entered and to do this the user selects an organisation name and WSDL document URL. From this the service and operation names can be retrieved. After selecting a service and operation the parameter structure is automatically created by parsing the parameters of the WSDL document. The ontology structure used here is simpler than WSDL as not all WSDL information needs to be saved. It is a very simple way of storing structure and content. The entered local ontology is then mapped to a global ontology. A user of the system then only needs to deal with and learn the global ontology and can enter a single set of parameters from which many Web Services can be called. If the global ontology does not contain enough parameters to cover a particular Web Service, new parameters can be entered and the

local ontology mapped to them. The system will then know what parameters to use when looking up a particular Web service or group of Web Services. When using the Web Services agents, we would present to the user a list of all possible global concepts for the Web Services in question. For example, consider all flight booking Web Services in a travel scenario. Some inputs may be unique to one Web Service and some may relate to more than one Web Service. The user would enter all required data and then let the program determine which parameters relate to which Web Service. We then construct the SOAP message dynamically taking all parameters into account. If the data being input is a complex type then the structure of it will be defined in the ontology.

5.3 Web Services Agents

There are three types of Web Service agent that have been built. An agent has been written that can call only one type of Web service. This is relatively simple. It does not use ontologies and can have hard-coded values except for the input parameter values. The reply from the Web Service can then be converted into a Java object for further processing. The second type of agent connects to a particular type of Web service, such as a stock quote Web Service. This needs to use the ontologies to construct the SOAP messages but can also map the reply message to a Java object. This is because we know the global ontology output parameters before writing the agent and so these can be directly mapped to Java fields. The local parameters can then be mapped to the global parameters dynamically allowing different Web Services to use the same Java object. This again means that the reply can be further processed by the system. The third type of agent can connect to any Web Service that can be defined by the ontology. This is useful if we find a Web Service and a specific

agent has not been written that can call it. However the reply from this agent cannot be processed further and can only be converted into XML to be displayed.

Having constructed the ontologies we can use the Web Services to retrieve explicit knowledge. This could be in the form of documents or information retrieved from a knowledge base. The explicit knowledge could also be retrieved from documents stored in the system in the Content Repository but maybe the Web Services will offer information that is updated more regularly or not available from another source. Although the number of Web Services at the moment are limited we expect their numbers to grow and cover a varied range of subjects. One likely scenario is where the Web Service acts as an interface to a search engine. The user would enter a number of keywords that describe information to be retrieved and the Web Service would return search results defining how relevant each document was to the input parameters. More than one agent may be looking up more than one Web Service so the agents would negotiate with each other to decide what the most relevant information would be. If cost was a factor then they may also negotiate on cost. Then only the most relevant documents would be retrieved.

Figure 5 gives a description of how JADE and the JWSDP (Java Web Services Development Pack) would be used to call a Web Service. From this diagram we can see that the IADE is passed a process activity in an XPDL (XML Process Definition Language) file. The IADE then creates an in-process agent, which is passed the XPDL file, as for the Communities of Practice agents. The in-process agent connects to the main container(s) where the permanent agents are running and passes the file to a Web Services agent. The ontology has been created from the WSDL document, so

the Web Services agent dynamically constructs a SOAP message using SAAJ and calls the Web Service.

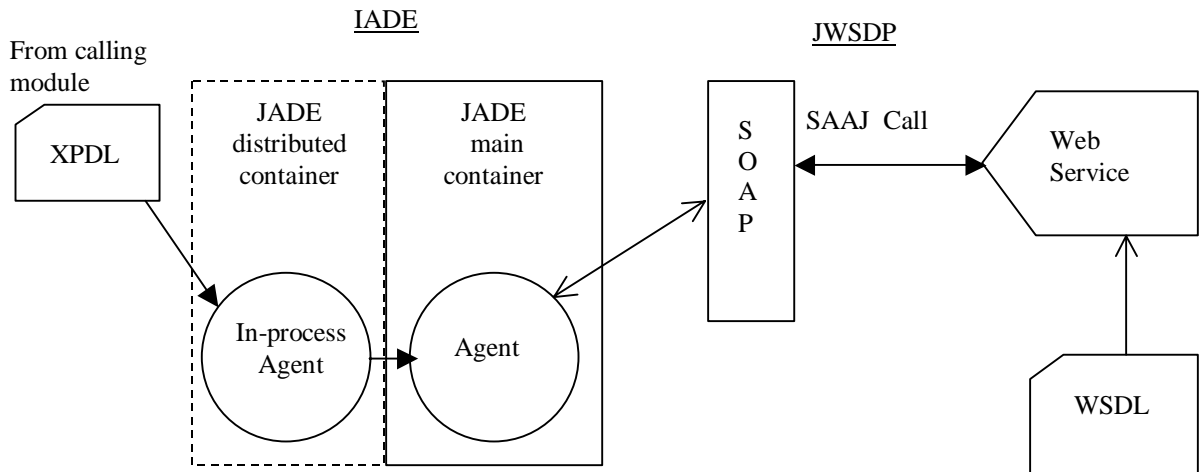


Figure 5. Diagram showing how JADE and JWSDP are used to call a Web Service.

5.4 Communities of Practice Agents

In our case this information would help with deciding the values that certain preconditions for a particular service should take. Having set the preconditions the Communities of Practice agents would be contacted to try and arrange for one of the local experts to carry out the service. It should be noted here that we need some way of deciding exactly which agents to use for any particular scenario. There has been much work done on resource location and [22] describes an agent-based resource location system. We will again suggest a simpler approach for a first implementation. We have a Classification Engine in our ICONS system that classifies documents to be stored in the system into different categories. For each document category we could also store the Web Service type and the Communities of Practice User and Expert

agent types that could be used to retrieve the information. A user would firstly inspect the documents stored in the system and depending on the category they belong to he would ask for particular types of Web Service to be used to retrieve other relevant information. On formulating the preconditions, he would get the User agent type and set the Expert agent types as parameters and initiate a Contract Net protocol between the Communities of Practice agents to negotiate the service. The User agent would receive the proposals from the Expert agents and would evaluate them. Eventually the evaluation may be done using DLV. The User agent would then reply to one Expert agent with an Accept Proposal protocol and a manual process would take place to create the contract for the service. As explained before, a manual process would be used to create a contract from which an expert would be selected. The user would then ask the expert(s) to carry out the service, who would use their experience and knowledge, communicating through email for example, to provide the service. There would also be a manual process allowing the user to override the suggestions of which agents to contact in the case where these agents were found not to be suitable. If a different type of agent was found to be more suitable then the structure that stores agents under each category could be updated to reflect this change. We can now look at a description of the IADE module as a whole in figure 6. It is called through the in-process interface by a calling module, which is one of the ICONS other modules, possibly related to workflow. The in-process interface then creates the in-process agents used to communicate with the permanent agents running in the system.

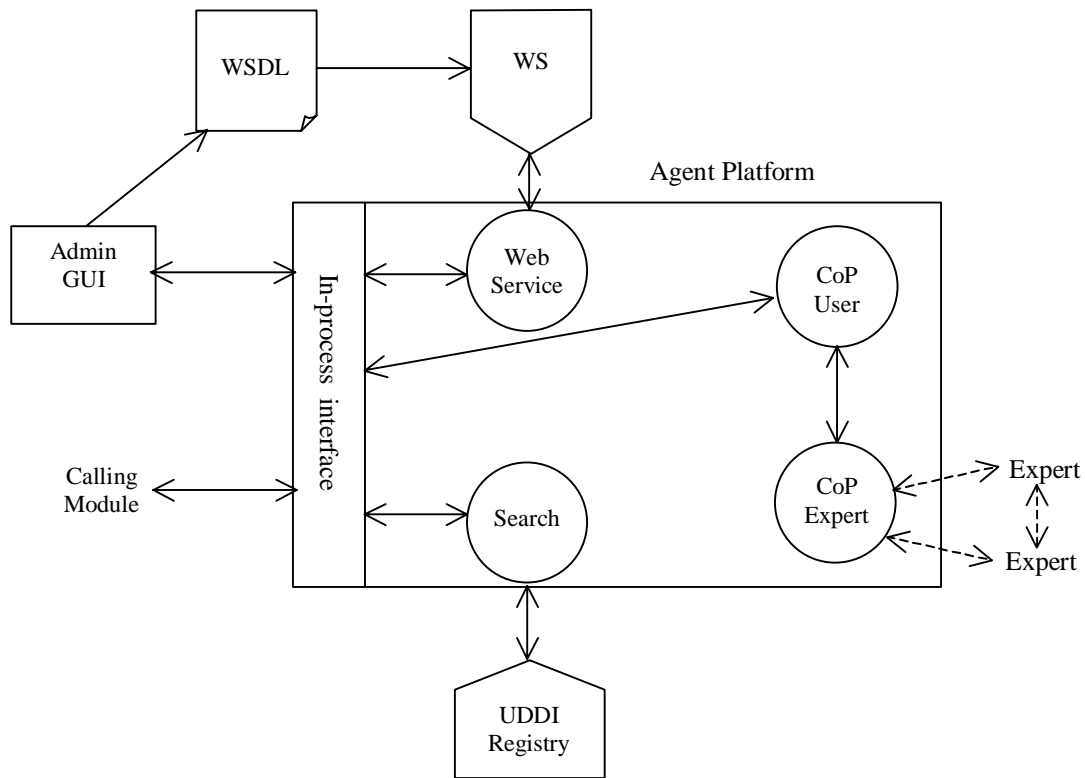


Figure 6. Diagram of the IADE module

6. Related Work

6.1 Communities of Practice Related Work

Various other systems have been developed that look at Communities of Practice or Web Services. Our system is simpler in nature but combines both CoPs and Web Services together in a knowledge management system. We can firstly consider other application of Communities of Practice. In [16], Roda et.al. discuss a knowledge management system using agents. Their agent-based system is designed to support the adoption of knowledge sharing practices within communities. Their system is called knowledge-intelligent conversational agents (K-InCA). The communities they are concerned with are broader than the sub-communities that CoPs would be, consisting of whole organisations and not just small groups formed within an organisation. The agents they use are to support the adoption and implementation of knowledge

management behaviours and so would be more sophisticated than the agents we have initially developed. The exact communication protocol is not described but user and expert agents are defined where an expert sends a proposal to a user for evaluation (as is done with our Contract Net protocol). [4] describes an agent-based system to develop a Communities of Practice in an e-commerce system. They describe how the agents can provide intelligent support for the system. They describe how at one end of the distribution chain are the manufacturers while at the other end there is the retailers. Historically the manufacturers controlled the market process, but there is a shift in the balance of power towards the retailers and agent-based systems help in this shift. By automating the selection process of the buyer, allowing searching and evaluation by agents for the best bargain, the consumer is gaining more say in the process. In this paper they describe a system where the different activities of supply-chain participants are viewed as resulting from the activities of Communities of Practice supported by multi-agent systems. In other words, a degree of close business relations is assumed. They also describe how personal service agents may group together and negotiate in order to get a better deal, for example, to buy in bulk. Instead of buyers and sellers we have users and experts, where the main power belongs with the users who, through agents, can automatically evaluate and select the appropriate expert for any task. We have not considered how users may group together and possibly ask for a service to be supplied to the group rather than individually. Finally [14] describes how virtual communities are proliferating through the internet. Communities of Practice address organisational and information system issues in these virtual communities. They describe how the community-based approach to knowledge management is considered as one of the most effective tools for knowledge creation and transfer. Transforming the traditional off-line CoPs into

on-line virtual communities will greatly improve their community scope, interaction efficiency and sharing of critical information and knowledge.

6.2 Web Services Related Work

Web Services are now gaining in popularity and some research has been done on them also. As described in section 4, McIlraith et. al. in [15] have developed a system using agents for automatic Web Service discovery, execution and composition. They use the semantic web markup language DAML-S to construct a knowledge base to define the Web Services concepts. They use the WSDL description to define the actual Web Services and DAML-S to describe what the service can do. They use semantic markup to define what the service provides and automatically manipulate this through agents and construct ontologies to allow automatic use of different Web Services. DAML-S is also described in [1]. However, the semantic web is still relatively immature and Web Services are currently not defined by DAML. As [2] points out, the Semantic Web community is defining standards for the representation of semantic Web Services, but this is still under development. They propose to improve the communication with Web Services through a speech-act based representation of conversations, where we have agent-like conversations with the Web Services. The semantic web seems to be the way to go but for the present we will search for Web Services only in UDDI registries and then require some manual selection. We needed to develop something that could be practically used in a current system and with this in mind we developed a very simple structure for the ontologies, that allows easy mapping between concepts and manipulation of the ontology, e.g. to enter and map new values. In our current system the ontologies are created automatically, but mapping between them must be done by a human. Future work could look at

automating this and [21] gives results for automatically creating ontologies for business-to-business Web Services. It describes ways of automatically providing syntactic similarities through a basic formula and semantic similarities through the lexical database *WordNet*. The approach they use is to translate between ontologies using a platform independent language based on XML (such as DAML). Agents create only relatively small local consensus ontologies (not very large global ones) to facilitate the discovery and understanding of Web Services that they use locally. They can then collaboratively develop a global consensus ontology at another stage. This idea of local ontologies is what is used in our system, where we simply map parameters of different ontologies to each other that are used in the local operation calls. Collaboration between agents for automatically discovering mappings has not been considered yet. Finally, providing further support for DAML-S, [6] builds ontologies for Web Services using this and also an agent communication language. They are then able to define different message types (e.g. directive or assertive message) and are able to query this using a query language.

7. Implementation

A brief description of the implementation will now be given. Two applications for the CoP have been developed. These are for ERDF project funding and to provide a service. In the ERDF funding application an expert is asked for answers to questions related to a project he is involved with, which are criteria for Structural Funds ERDF funding. Answers will include tacit knowledge tailored to specific questions or criteria. A user asks experts for answers with a certain score. The experts store their standard answers in a file and the expert agent automatically returns these, which are then evaluated. If the expert passes the score he is asked for more specific information

to meet the particular criteria of the user. The human expert needs to provide this. The user can then select projects for possible ERDF funding. The second application is for an expert to provide a service to a user. The user forms a profile of the service he requires and sends it to specified expert agents through a user agent. Each expert agent stores a profile of the service he can provide, in the form of an XML file. This is checked against the user profile to determine if it meets the preconditions. If it does it is returned to the user agent, which select the best profile(s) among all returned. The selected expert agents are then informed and asked for contracts. The expert agents then pause waiting for the human experts to complete and return the contracts, which are then returned to the user agent. The user agent then returns them to the human user who can inspect the contracts and select the one he prefers to provide the service.

It is the intention of the system that the user would firstly use Web Services to formulate initial criteria for the service required. There are not any Structural Funds Web Services at the moment, so the Web Services functionality has been tested separately from the CoP functionality in the current system. We have tested various Web Services, including stock quote and statistical Web Services. The diagrams below are of the GUI used to test the system. There is a test panel used to test the different agents, which we expect will be removed from the final ICONS system. There are also panels for admin functions such as agent admin in Figure 7 (this is the same as the RMA GUI of the JADE platform with an admin password added and tools menu removed) and creating properties files for the agents and XML search and services profiles. Figure 8 shows what the returned search information looks like. This is then saved in the permanent search tables from where it can be retrieved to create ontologies. This manual process involves the user looking up and testing the Web

Service to see if it matches his needs. Only UDDI registries are queried so only a limited number of Web Services are returned at the moment. The manual process is not too time consuming and in fact it is quite difficult currently to find a suitable Web Service through the UDDI registries. There are also various search criteria that can be used to narrow the search. However, as the number of registered Web Services grows we will need to make this search process more sophisticated. This will all rely however on the use of a semantic markup language and ontologies to define the Web Services on the internet. Figure 9 shows the panel used to create an ontology for a stock quote Web Service. Figure 10 then shows this ontology being mapped to a global ontology. In this panel the global ontology may contain some redundant values left over from the local ontology it is created from. These are not relevant and do no harm. The user then only enters a single set of values for the global ontology. Figure 11 then shows parameters being entered to test one of the Web Services. Integration with the rest of the system is still ongoing. Ideas like the connection with the Classification Engine are still just ideas and have not been implemented. A really autonomous search agent has also not been implemented.

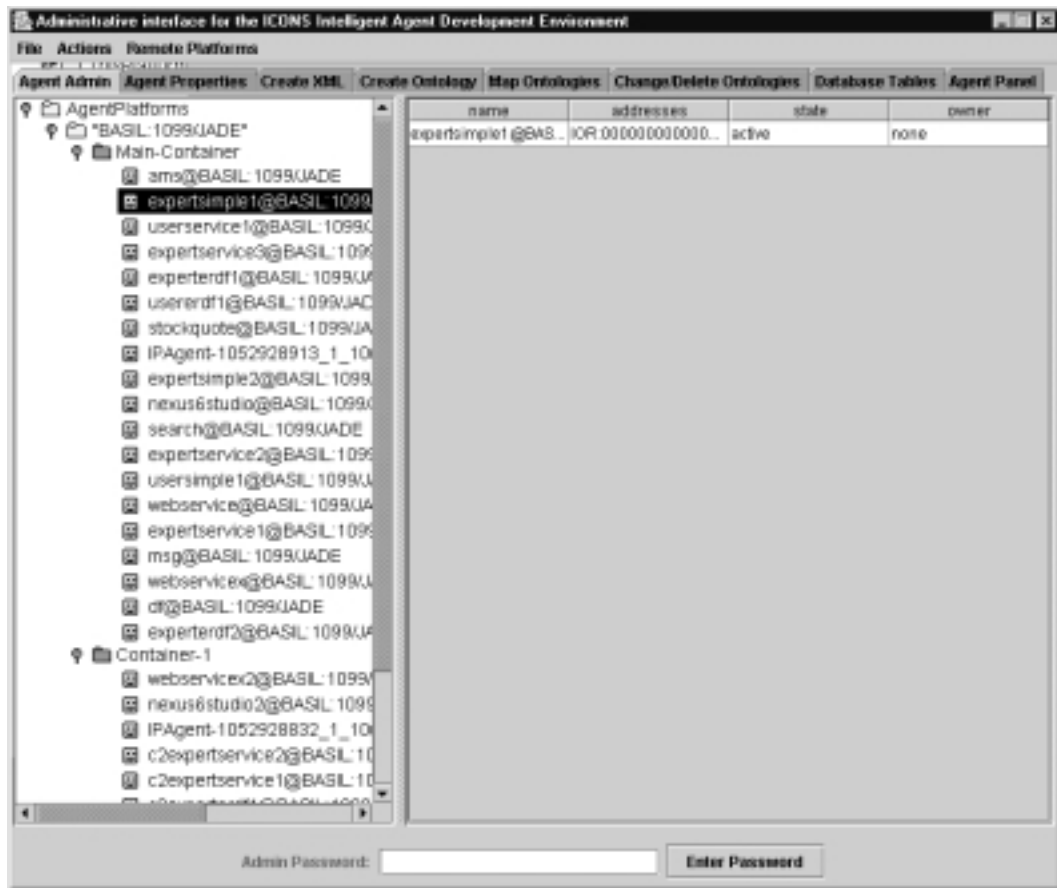


Figure 7. Diagram of the agent admin panel. This is the same as the JADE RMA GUI with an admin password added and tools menu removed

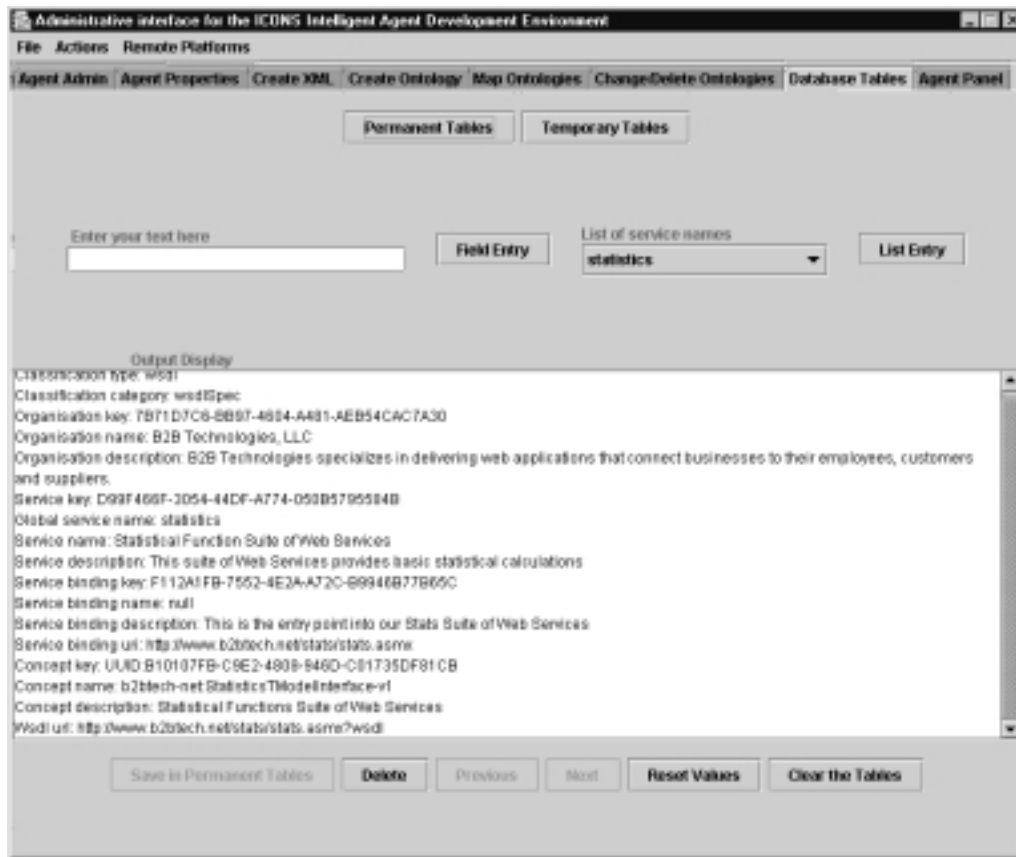


Figure 8. Diagram of the database panel. Service binding uri and Wsdl url are the most important entries

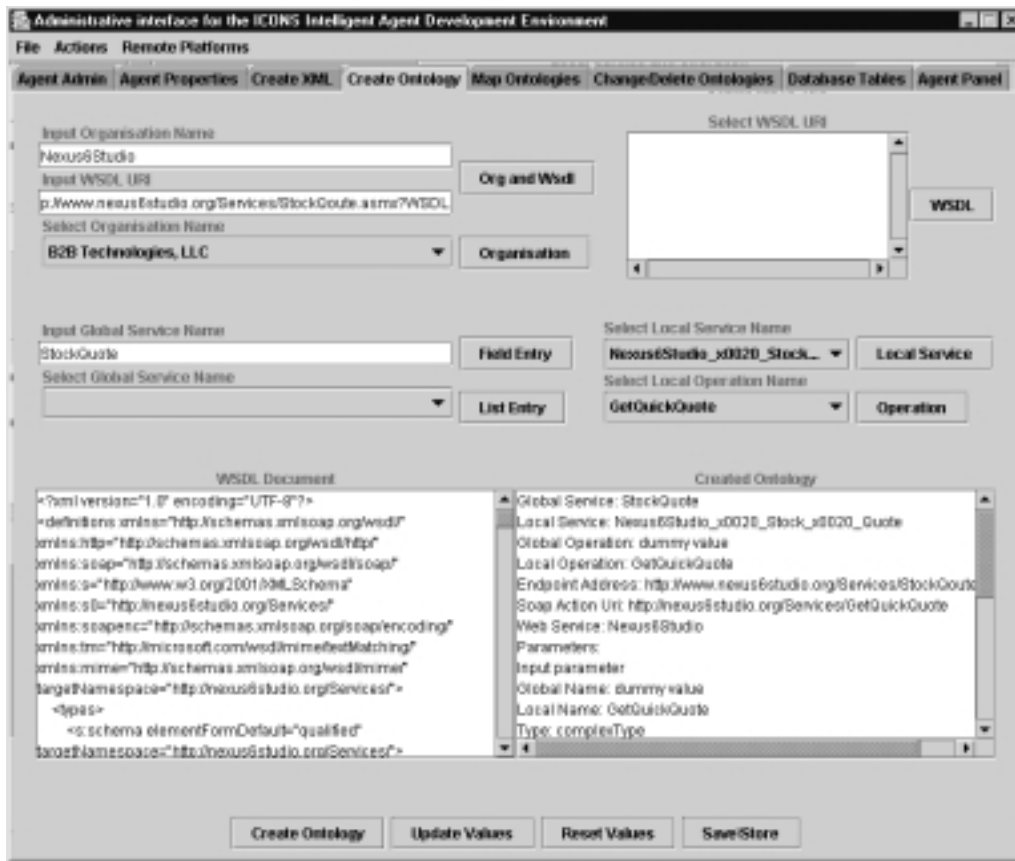


Figure 9. Diagram of the create ontology panel. Created ontology is on the right. Dummy values are replaced when the ontology is saved or mapped



Figure 10. Diagram of the ontology being mapped to a global ontology. Global ontology is on the left and the ontology just created is on the right

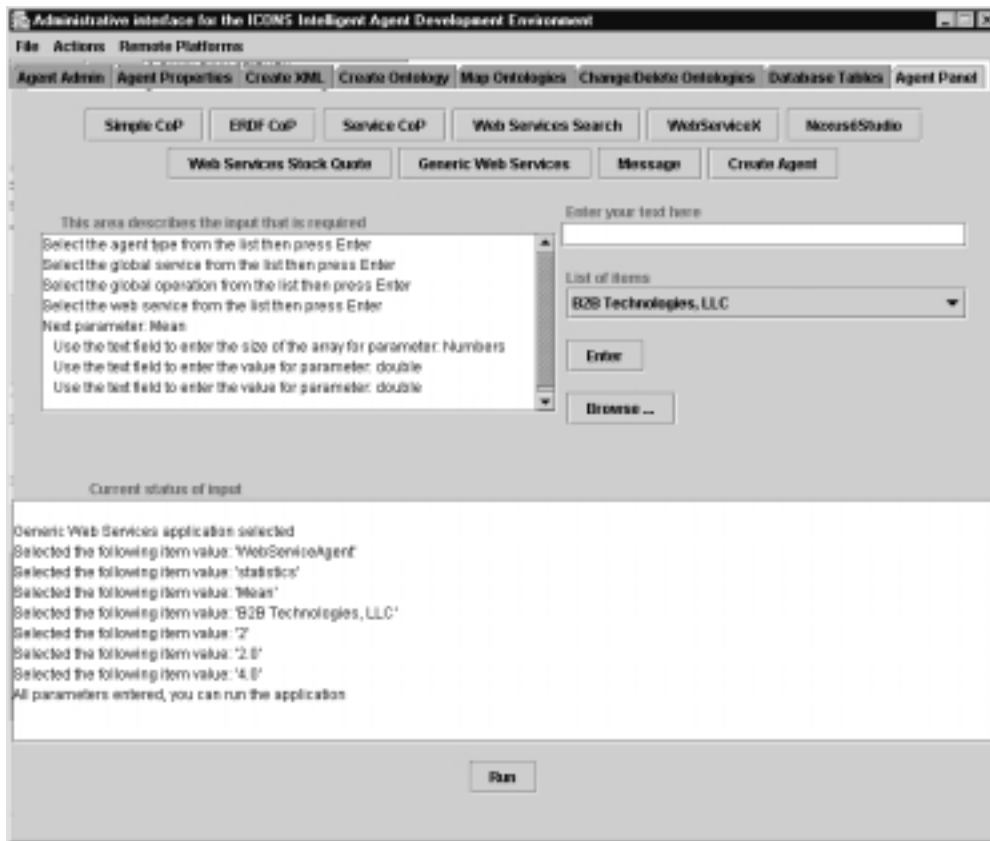


Figure 11. Diagram of the test panel. Values are entered to call a statistical Web Service

8. Conclusions

We have shown how some new technologies may be used together to provide part of a knowledge-based system for a user. Through the use of XML and ontologies this system will allow the integration of information from different sources. Setting up the system will require a certain amount of manual input but then intelligence coded into the agents should allow for a mainly automatic action. The system notes the availability of both explicit and tacit knowledge and intends to use both types to provide a service for a user. The system is to be used as part of the ICONS [9] system as an Intelligent Agent Development Environment (IADE) module. The ICONS system as a whole will provide much more functionality to the user. To be integrated with ICONS the system we need to integrate with a Content Repository, Classification Engine and a Workflow Manager. The ICONS project has reached its

implementation stage and so these components are currently being developed. An advanced GUI has been written to allow input of ontologies and other input files and also to run the agents, though ICONS will call the agents through a Java api. In the ICONS project we hope to use the system in the area of a Structural Fund project with a possible application being risk analysis of infrastructure investment projects. As the system is in its early stages there is much scope for extending it, in particular in the area of automating the Web Services search process and ontology construction.

Acknowledgements

This work was funded by the ICONS project. The Intelligent CONTENT Management System (ICONS) project is realised within the European Commission Fifth Framework Programme, User-friendly information society IST (see www.cordis.lu/ist). The project, with the identifier IST-2001-32429, addresses the objectives of the action line IST2001 - II.1.2: Knowledge management.

References

[1] Anupriya Ankolekar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, Drew McDermott, mDavid Martin, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne and Katia Sycara, DAML-S: Web Services Description for the Semantic Web, in Proceedings of the 1st International Semantic Web Conference (ISWC 02), 2002.

[2] L. Ardissono, A. Goy and G. Petrone, Web technologies: Enabling Conversations with Web Services, Proceedings of the second joint international conference on Autonomous agents and multiagent systems, July 2003, pp. 819-826.

[3] Buccafurri, F., Leone, N. and Rullo, P., Disjunctive Ordered Logics: Semantics and Expressiveness, Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR '98), 1998.

[4] Marcel Cohen and Kostas Stathis, Strategic change stemming from e-commerce: implications of multi-agent systems on the supply chain, Strategic Change, Vol. 10, 2001, pp. 139-149.

[5] FIPA, Foundation for Intelligent Physical Agents, <http://www.fipa.org/>

[6] Nicholas Gibbins, Stephen Harris and Nigel Shadbolt, Using the Semantic Web: Agent-based Semantic Web Services, Proceedings of the twelfth international conference on World Wide Web, May 2003, pp. 710-717.

[7] Hahn J and Subramani MR, A Framework of Knowledge Management Systems: Issues and Challenges for Theory and Practice, 21st International Conference on Information Systems (ICIS 2000), Brisbane, Australia, 2000.

[8] Huberman, B.A. and Hogg T., Communities of Practice: Performance and Evolution, Computational and Mathematical Organization Theory (1), 1995, pp.73-92.

[9] Icons, <http://www.icons.rodan.pl/>.

[10] JADE, <http://jade.csel.it/>.

[11] Java, <http://java.sun.com/>.

[12] Jennings, N. R., On agent-based software engineering, *Artificial Intelligence* (117), 2000, pp.277-296.

[13] Kimble, C., Hildreth, P. and Wright, P., *Communities of Practice: Going Virtual*, Chapter 13 in *Knowledge Management and Business Model Innovation*, Idea Group Publishing, Hershey (USA)/London (UK), 2001, pp 220 – 234. ISBN 1 878289 98 5.

[14] J. Koh and Y.-G. Kim, Knowledge sharing in virtual communities: an e-business perspective, *Expert Systems with Applications*, ??, 2003.

[15] Sheila A. McIlraith, Tran Cao Son and Honglei Zeng, *Semantic Web Services*, In *IEEE Intelligent Systems (Special Issue on the Semantic Web)*, March/April, 2001, pp. 46-53.

[16] Claudia Roda, Albert Angehrn, Thierry Nabeth and Liana Razmerita, Using conversational agents to support the adoption of knowledge sharing practices, *Interacting with Computers*, Vol. 15, 2003, pp. 57-89.

[17] Systinet, Introduction to Web Services, Whitepaper, http://www.systinet.com/resources/white_papers.

[18] The IST-2001-32429 ICONS Consortium, The Structural Fund Project Knowledge Portal Conceptual Design, Deliverable 35, www.icons.rodan.pl, Dec. 2002.

[19] Wenger E, McDermott R and Snyder W.M., Cultivating Communities of Practice: A Guide to Managing Knowledge, Harvard Business School Press, Boston, Massachusetts, 2002.

[20] WfMC, WorkFlow Management Coalition, WorkFlow Process Definition Interface – XML Process Definition Language, WfMC-TC-1025, draft 0.03a, May 2001.

[21] Andrew Williams, Anand Padmanabhan and M. Brian Blake, Local Consensus Ontologies for B2B-Oriented Service Composition, Proceedings of the second international joint conference on Autonomous Agents and multiagent systems, July 2003, pp. 647-654.

[22] Wills G, Alani H, Ashri R, Crowder R, Kalfoglou Y and Kim S, Design Issues for Agent-based Resource Locator Systems, In Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management (PAKM'02), Vienna, Austria, December 2002.

[23] WSIF, <http://ws.apache.org/wsif/index.html>.