ADEPT* - Advanced Decision Environment for Process Tasks: Overview and Architecture

J.L.Alty‡, D.Griffiths†, N.R.Jennings‡‡, E.H.Mamdani‡‡,
A.Struthers††, M.E.Wiegand†

† BT Laboratories
††ICI Engineering Technology
‡ Loughborough University of Technology
‡‡Queen Mary and Westfield College

Abstract

This paper provides an introduction to project ADEPT (Advanced Decision Environment for Process Tasks). The project is researching both the technology and the methods that are needed to improve the way information is gathered, managed, distributed and presented to people in key business functions and operations. The paper presents a first-level breakdown of the central architecture concepts that have been emerging during the first six months of the ADEPT project. The use of autonomous agents to provide information services is explored and a language for their communication and representation developed. The work presented here represents only the preliminary steps towards the final architecture and is likely to undergo significant refinement throughout the project life-cycle.

1.0 Introduction

Company managers make informed decisions based on a combination of judgement and information from marketing, sales, research, development, manufacturing and finance departments. Ideally, all relevant information should be brought together before judgement is exercised. Requesting pertinent and consistent information across a large company is a complex and time consuming process. Changes to information made in one department can have repercussions throughout a company often invalidating previous

---

* ADEPT is a collaborative project under the DTI/EPSRC Intelligent Systems Integration Programme (ISIP). The project partners are BT Laboratories (lead), ICI Engineering Technology, Loughborough University of Technology, and Queen Mary and Westfield College.

Acknowledgement: The Service Description Language described in this paper has been developed by J. Stein of Queen Mary and Westfield College.

decisions. For these reasons, the ADEPT project is researching both the technology and the methods that are needed to improve the way information is gathered, managed, distributed and presented to people in key business functions and operations. The project is creating a *Concurrent Information Environment*. The environment will be used to demonstrate that requests for relevant and timely information can be made proactively by business systems in advance, and that the systems themselves can be responsible for maintaining the consistency of the decisions that are made with respect to this information.

The inherent distribution of data, problem solving capabilities, and responsibilities, coupled with the desire to maintain the integrity of the existing organisational structure, meant that an agent-based solution was adopted. Agents are goal-oriented entities which are able to solve autonomously particular problems and be responsive to changes in their environment. Whilst engaged in their problem solving they may interact with other agents using an agent communication language [1].

A short technical overview outlines the major themes of the project: information infrastructure issues, information management and information presentation. An outline of the architecture that provides the logical model for the negotiation of information and services is given. Finally, languages for representation and communication are described.

2.0 Issues of Information Integration

The technology research activities address three areas that support information integration: Information Infrastructure (section 2.1), Information Management (section 2.2), and Information Presentation (section 2.3). Specific technology research issues relate to how information is negotiated for, transparently accessed over an heterogeneous network and reasoned over. The research path is dictated by requirements emanating from the work of the industrial partners.

2.1 Information Infrastructure

The information infrastructure issues relate to the provision of an open distributed environment that supports the coordination and cooperation required to realise a complex business process. The environment must have the ability to provide relevant information in a timely manner. At run time the environment will determine what information should be sent to whom and when. There must be a series of mechanisms to realise and maintain the dependencies and interrelationships specified in the information management definitions. It is essential that application services residing in a number of different physical and logical contexts can be shared and accessed transparently.

The environment must provide support for agreement making, negotiation [2] and conflict management [3]. This involves indicating where and when such agreements are necessary, identifying who needs to be party to the agreement, and providing support for the ensuing negotiation process.

2.2 Information Management

Information Management issues relate to the ways in which processes can be modelled and controlled automatically. A main tenet of this research is that the model of information flow in the work process (information viewpoint [4]) should be separated from the model of responsibility (enterprise viewpoint) for the various stages of the work process. This ensures a separation of information and organisational issues that leads to robust information management.

Information and process management is controlled through the maintenance of 'audit trails' and automatic consistency checking of actions and decisions involved in the process. Management functions such as planning and scheduling enable the environment to reschedule the work process and redirect responsibility automatically in the event of some problem occurring.

The integration of temporal information will enable the environment to make inferences involving time, such as predicting the time to complete the work process, or advising of delays beforehand.

2.3 Information Presentation

Information Presentation issues relate to presentation requirements of effective display techniques for information from disparate sources, in different formats, that is propagated across heterogeneous network environments.

Presentation techniques include the use of interface *metaphors* [5] to replace (or supplement) the conventional object references in computing systems, of data files, programs, etc. Spatial alternatives such as office or desktop metaphors need to be supported. Information fusion is also seen as a presentation issue. This relates to how the added value that arises from the processing of information is presented to a human or any decision point in a process.

Furthermore, multimedia technology [6] is to be supported by the environment. This technology will place a requirement for coordinating information presentation on the appropriate media available in the environment.

3.0 System and Agent Architecture

3.1 System Architecture

The top-level concept of the architecture is of an infrastructure consisting of numerous *agents*. An agent controls and manages the provision of a number of *services* to other agents on the infrastructure. The top-level architecture concept is depicted in figure 1.

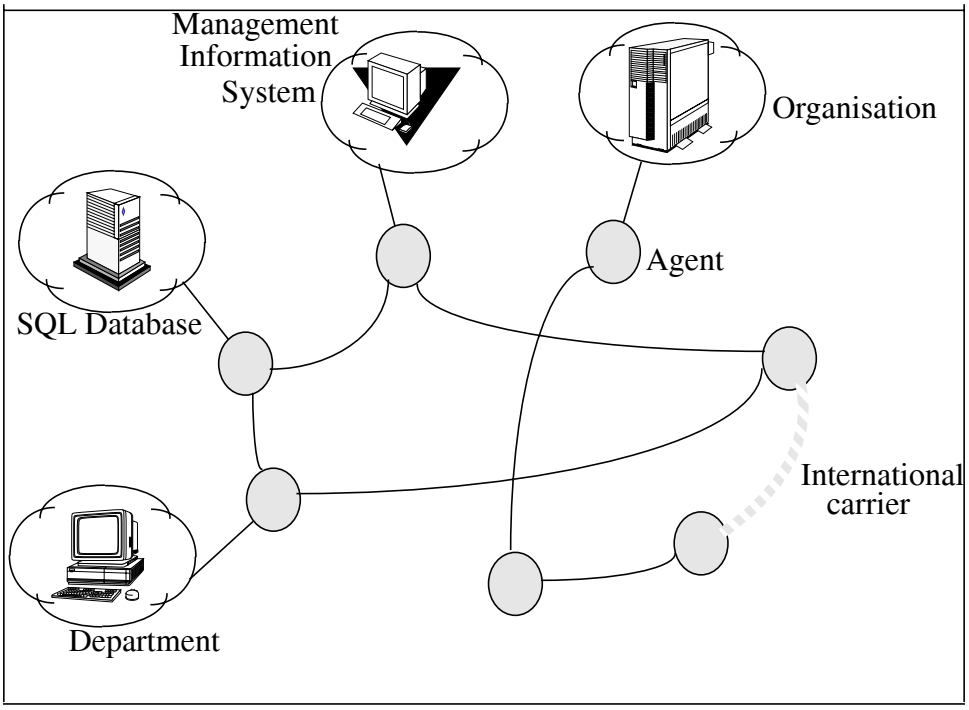

FIGURE 1. Top-Level System Architecture Concept

The agents are able to communicate, and negotiate with each other for the provision of services. Services provided by agents can range from the provision of information and data, or the implementation of some concurrent task within a business process, through to the setting up of international communications on demand (by negotiating with a carrier), or the provision of presentation functionality. An agent on the infrastructure is characterised by the services that it provides.

There are three high-level requirements of the ADEPT architecture:

1. *autonomy* - the local organisation should be empowered to define how it (they) will perform their local tasks and processes;
2. *concurrency* - the tasks and the services must be able to be run concurrently, with the interactions monitored and managed automatically;
3. *migration* - agents will need to define new services incrementally, without the need to redesign the entire distributed system.

3.2 Agent Organisation

The basic building block of the architecture is shown in figure 2. This basic unit comprises an agent and the *tasks* that are under its control. This unit provides one or more services all of which are under the control of the agent. In a “free market” model each of these units

would be able to negotiate with any other agent when supplying a service. Furthermore, an agent could negotiate with any other agent in order to enhance it's services or to create new services.

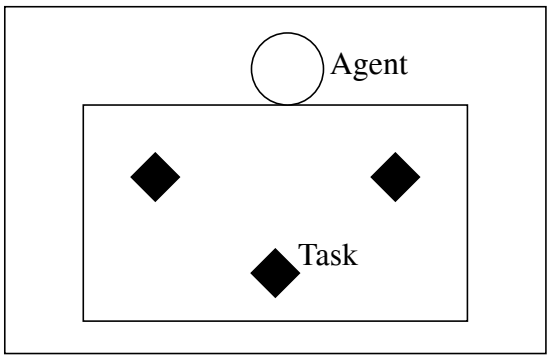

FIGURE 2. Basic Unit of Organisation

The free market agent model is very simplistic, and not practical in a commercial environment. Commercial environments are founded on organisational models where an organisation is logically divided into a collection of services. The ADEPT architecture draws upon this principle to group services and tasks where it makes pragmatic sense. A grouping of tasks and services is known collectively as an *agency.*

Agents may communicate with other agents in two ways: loosely coupled and tightly coupled. In a loosely coupled interaction each agent has an equal status, no one agent is controlling another. In tightly coupled mode, one agent is a controlling agent and the other agents have restricted access from agents outside of the agency. Even so these agents still have a large degree of autonomy. The relationship between an agent and a task is always close coupled; see figure 3.

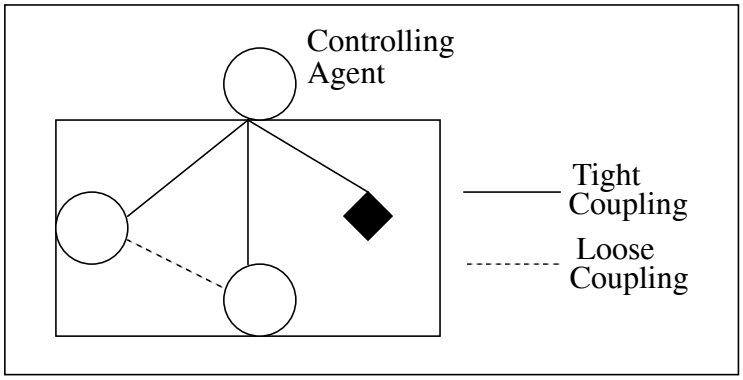

FIGURE 3. Coupling of Services and Tasks within an Agency

In this organisational model, servant agents reside in an agency. These servant agents are loosely coupled to each other but tightly coupled to the controlling agent. The controlling agent provides access to the world outside its agency. Agents within an agency may only negotiate with external agents through the controlling agent. An agent can have both the role of a controlling agent and a servant agent. A controlling agent will normally be a loosely coupled agent in a higher level agency. Agents in an agency will usually be the controlling agents of lower level agencies. This leads to an hierarchical organisation of agencies reflecting the logical structure.

In the above description, two extreme levels of agent interaction are outlined: negotiation/ communication and task control. In fact it will be possible to have a spectrum of interaction between these extremes. For example a loosely coupled agent will have a complete set of agent communication services, the tasks in an agency will have the minimal set of communication services that allow them to merely respond to an agent. Other agents may have partial sets of agent services.

To extend this model further, we consider how a number of agencies can be used in providing a service. The agent requesting the service is designated as the controlling agent and then a set of agents from different agencies can be selected to form a *virtual agency*; see figure 4. Note that this model reflects the principle of concurrent engineering whereby agents from different parts of a logical organisation may cooperate in the provision of some specific service.

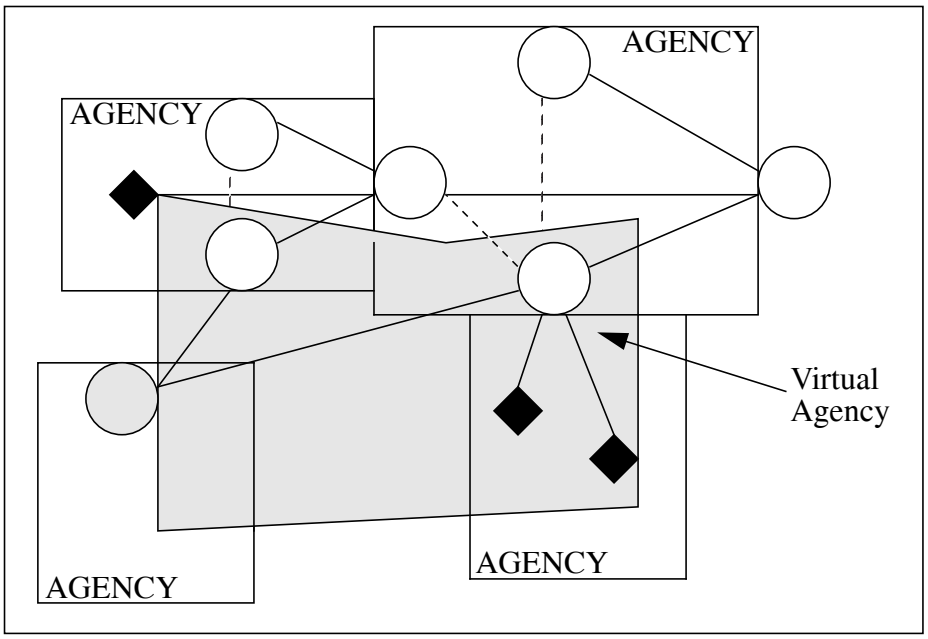

FIGURE 4. A Virtual Agency

3.3 Agent Architecture

The top-level agent architecture is shown in figure 5. This figure illustrates that information on the services provided is contained within the agent, as well as the capabilities to negotiate about its services, or about the services requested from other agents. The acquaintance model enables the agent to build a picture of information as to which agents provided which services in the past and whether contractual agreements were met, etc.

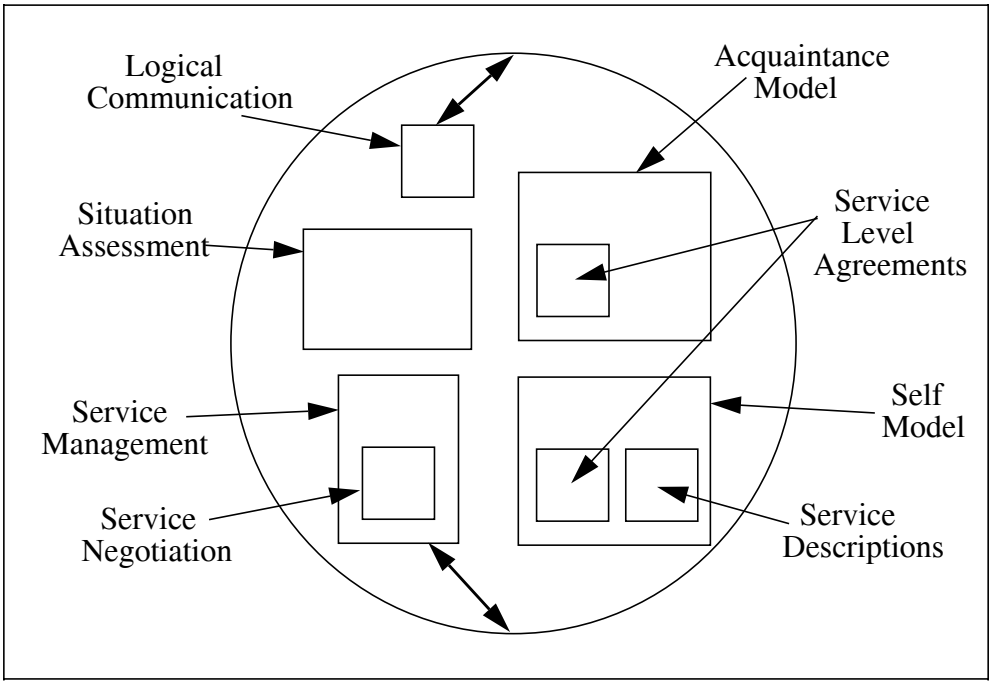

FIGURE 5. Top-Level Agent Architecture

The agent has two external interfaces: one to the rest of the community (through the logical communication module) and one to its local agency (through the service management module). The former enables the agent to communicate with other agents in the system at a high-level and using the primitives of the inter-agent negotiation language (see section 4.2) - at this level issues concerned with the physical infrastructure are transparent. The latter is concerned with managing the services that the agent is executing in its agency (this is achieved using the intra-agent management language) - this includes: scheduling concurrent services, starting new services, passing information onto relevant services, and monitoring the status of active services.

The agent also contains a situation assessment module, a service negotiation module and a service management module. The situation assessment module serves as a link and balancer between an agent's two primary roles - that of being an individual and that of being part of a social community. The service negotiation module is invoked when the situation assessment module indicates that a contract should be established with some

external agent for providing a particular service. This module handles the negotiation to set up a service level agreement (SLA). The service management module deals with SLAs which have been established - ensuring that terms and conditions are honoured, that appropriate levels of reporting take place, and that the SLA is cleanly finished.

All of the functional modules within the agent make use of the self and acquaintance models - the self model maintains a representation about the capabilities and status of the local agency, whilst the acquaintance model maintain a similar record of the other key agents in the system [7].

4.0 Representation and Communication Languages

4.1 Service Description Language

One of the key roles of an ADEPT agent is that it provides application services. A service can be requested (or negotiated for) by another agent, and typically this may result in the transfer of information between the agents. Agents must contain the definitions of how to provide each of the services that it offers. Accordingly, a *description* for each service is represented in the agent. A service description is like a "recipe" made up of two types of primitives:

1. tasks - that are executed entirely within the control of the agent;
2. services - provided by other agents over the infrastructure.

Tasks and services may be initiated concurrently, and the interaction and communication between them is managed automatically by constraints in the service description and by maintaining an "audit trail" of agent negotiations. Note that services in turn may be defined recursively in terms of services from other agents, but that eventually the definitions will be solely in terms of tasks. In more detail, a service is defined by the following features:

1. Each service has a unique **name**.
2. Each service has a **guard**. A guard declares information which is required in order to launch the service. There may be also constraints expressed, such as, a service may be launched only when some value x is available and is greater than 5, etc.
3. Each service may have **assumptions**. An assumption may impose further constraints on information, or assign default values (which are then believed, but perhaps not yet known).
4. Each service has a **body** which describes how the service can be executed.

To give an example, that part of an agent's self model which contains the service descriptions may be instantiated in the following manner:

```
(Service
    :Name PriceForecast
    :Guard (Region, ProductionCapacity,
                CapacityCompetition, Demand,
                Experience)
    :Assumption (SalesLevel :Default 100
                :Obtain)
    :Body (Sequence
            (Parallel
              (Sequence CreateTable,
                        SetYears (94..96))
              Calculate (Region,
                        ProductionCapacity,
                        CapacityCompetition,
                        Demand,
                        Experience))
          StoreInTable))
```

This states that a price forecast is produced based on information about a region, the production capacity of the company and the capacity of the competition, the demand specified by a client, and the experience of the company which is fixed in the service's *guard*. In general, this information will not be available automatically and has to be provided on demand. This will be done by looking up an agent's self and acquaintance model. Most of the information which is not given beforehand might be obtained via launching appropriate services as indicated by a "provided-by" slot. After checking this initial condition (guard), in order to execute the appropriate calculations, the agent will *assume* that the sales level (in relation to the specified demand) be 100%. Albeit, in order to justify its calculation, the agent will have to launch a service to *obtain* the definite value of the sales level. When this has been checked, the agent will initiate the service execution, i.e. launch the *body*.

It should be noted that none of the services in the body needs to be executed by the agent itself. It is likely that the parallel part of the body will be provided by different agents. In general, each of the services in the body will be represented somewhere in the agent's self or acquaintance model. In fact, what the body describes is that a price forecast is provided as a short sequence of services. The first service is a parallel service which is followed by the service StoreInTable. The parallel service describes the creation of a table in which the years 94 to 96 are initialised. At the same time (in parallel), an agent performs some calculation. When this has been provided successfully, then the results may be stored in the table.

4.2 Agent Communication

Following on from the basic organisational principles of agents, there is a need to define a language that enables the agents to communicate and negotiate. Furthermore when two or more agents are communicating they must share a common view of the problem domain; a *shared ontology* [8]. In this section, we explore the communication primitives.

The model developed here is motivated by consideration of the negotiation that would be required between two peer level agents each representing their own respective agencies. The following *Generic Service/Information Negotiation Model* has been developed:

1. `Find/Locate` (a service or piece of information) - make contact.

2. `Ask/Explain` (what is the form or nature of the service or information) - get the details.

3. `Establish/Negotiate` (setting up a service or receiving some information) - agree what is to be done.

4. `Execute/Review/Inform` (perform the service or deliver the information, with the ability to review progress and inform when necessary) - do what you agreed.

5. `Terminate`.

It is important that this model should apply to both explicit service requests as well as the more general, implicit service requirement of finding, locating and managing information within a business. This model was then developed into a set of specific primitives that could be used as a basis to explore further the precise nature of the negotiation semantics. These primitives are now described using the same numbering order in the model above:

1. CAN (YOU/ANYONE) DO<a service *s*>
   CAN (YOU/ANYONE) PROVIDE<information *i*)
   -->> return list of service/information providers

   These primitives would be equally applicable in a multi-cast situation involving department or enterprise based agents and location brokers, as well as a contract net type of broadcast request for the service. Various degrees of sophistication could be imagined in terms of the details associated with the location of the service provider dependent upon the type of models that are developed for agents themselves. These primitives can be summarised as saying “WHO” can do this and “WHERE” they are.

2. SERVICE DETAILS<ask agent *a*, regarding service *s*>
   INFORMATION DETAILS<ask agent *a*, regarding information *i*>
   -->> return list of <Inputs required>, <Outputs Given>,
   <Optional Resource (“time” /”cost” /...) estimate>

These primitives let you establish exactly what the service consists of in more detail. It can be summarised as saying "WHAT" can be done. As no global common language is assumed, an essential additional primitive here will be "EXPLAIN <ontology/term>" which will be needed to allow agents to handle new terms.

3. PROPOSE<agent *a*, service *s*, conditions c, optional rationale r>
   MODIFY
   REJECT
   ACCEPT
   -->> return <agreement (working conventions, reporting level, penalties, etc.)>

   These are the basic primitives that allow agents to agree how a service is to be provided. The conditions will be split into "mandatory" and "desirable". The optional rationale and details of service provision should provide agents with a mechanism to break deadlocks arising from (unnecessary) mandatory requirements, or provide more novel/alternative forms of service provision. This should allow much more sophisticated forms of negotiation processes to be developed/researched. The key thing to note here is that accepting a proposal, after any modifications (and perhaps nested asking/explanation), results in a specific service level agreement that can then be scheduled to be carried out.

4. INVOKE<agent *a*, service *s*, agreement c)
   INFORM (i.e. notify)<agent *a*, mode of acknowledgement *m*>
   QUERY/INSPECT<service *s*>
   REVIEW<services delivered *d*>

   Services are invoked with respect to some prior agreement, after which the progress can be queried or formally reviewed according to the reporting levels in the service level agreement. If it is necessary for the agents to inform each other of events (unforeseen or otherwise) then this can be done with the mode of acknowledgement set to "none", or "need acknowledgement" or "renegotiate". This is a critical primitive that will allow the whole negotiation process to become nested if necessary.

5. TERMINATE<status *s*, optional reason *r*>

   When a service terminates it will either have satisfied its "success" or "delivery" criteria set out in the service level agreement or not. If not, an optional reason should be given for termination.

5.0 Conclusions

This paper has presented a brief insight into the ADEPT project. ADEPT proposes the use of autonomous agents to provide information services. The agents are able to negotiate for services to provide a flexible and dynamic process management environment. The advantage of this approach is the suitability to global interworking and proactive (anticipatory) information management.

A first-level breakdown of the central architecture concepts that have been emerging during the first six months of the ADEPT project have been presented. This paper has put forward a number of terms and definitions and architecture concepts to be adopted within the project as a way of integrating the various research and application work areas.

The developing system architecture and the research paths are directed by requirements emanating from a number of applications drawn from the domains of the industrial partners. BT are investigating the use of the ADEPT environment to support a number of concurrent business processes, such as in the generation of customer quotes, from accepting customer requirements through to producing appropriate network designs. ICI Engineering Technology are applying the ADEPT concepts to supporting the safety procedures that must be followed in industrial plant design. Another application is considering the information support process for advising on financial and marketing strategy.

The work presented here represents the preliminary steps towards the final architecture, and these ideas are likely to undergo significant refinement throughout the project life-cycle.

6.0 References

[1] M. J. Wooldridge & N. R. Jennings (1995) “Intelligent Agents: Theory and Practice” Knowledge Engineering Review.

[2] B. Laasri, H. Laasri, S. Lander and V. R. Lesser (1992) “A Generic Model of Intelligent Negotiating Agents” Journal of Intelligent & Cooperative Information Systems” 1(2) 291-318.

[3] M. Klein (1991) “Supporting Conflict Resolution in Cooperative Design Systems” IEEE Trans. on Systems Man and Cybernetics 21 (6) 1379-1390.

[4] Recommendation X.901, ISO/IEC 10746-1 “Basic Reference Model of Open Distributed Processing - Part 1: Overview and Guide to Use”.

[5] J. M. Carroll and R. L. Mack (1985) “Metaphor, Computer Systems and Active Learning” Int. Journal of Man-Machine Studies 22 39-57.

[6] J. L. Alty (1991) “Multimedia: What is it and how do we exploit it?” Keynote Address to HCI’91, in D. Diaper and R. Winder (eds.) Cambridge University Press 31-44.

[7] N. R. Jennings, E. H. Mamdani, I. Laresgoiti, J. Perez & J. Corera (1992) “GRATE: A general framework for cooperative problem solving” IEE-BCS Journal of Intelligent Systems Engineering 1(2) 102-114.

[8] T. R. Gruber (1991) “The Role of Common Ontology in Achieving Sharable Reusable Knowledge Bases” Proc. of Second International Conf. on Principles of Knowledge Representation and Reasoning, San Mate, CA.