



UNIVERSITY OF  
CALGARY

# ***Tutorial Report***

**“Using Intelligent Agents to Manage Fuzzy  
Business Processes”**

**SENG 609.22  
Agent Based Software Engineering**

**Instructor: Dr. B.H. Far**

**By:  
Nora Houari**

Dec. 2002-ENEL- University of Calgary-

## Table of Contents

<b>1. INTRODUCTION</b> .....	<b>3</b>
<b>2. RESEARCH APPROACH: Use of IA Framework</b> .....	<b>6</b>
2.1. Intelligent Model-Based Synthesis Agent System.....	6
2.2. Architecture of an IMBS Agent.....	10
<b>3. FUZZY APPROACH</b> .....	<b>13</b>
3.1. FUZZY SCHEDULING APPROACH .....	15
3.2. Fuzzy Module-Based synthesis Approach.....	16
<b>4. SYSTEM IMPLEMENTATION</b> .....	<b>18</b>
<b>5. OVERVIEW AND EVALUATION OF THE RESEARCH PAPER</b> .....	<b>18</b>
<b>6. CONCLUSION</b> .....	<b>19</b>
<b>7. REFERENCE</b> .....	<b>20</b>
<b>8. ACKNOWLEDGEMENT</b> .....	<b>20</b>

## □ Abstract

Businesses are undergoing a major paradigm shift, moving from traditional management into a world of agile organizations and processes. An agile corporation should be able to rapidly respond to market changes. For this reason, corporations have been seeking to develop numerous information technology (IT) systems to assist with the management of their business processes. Agent technology looks set to radically alter not only in which computers are interacted but also the way complex processes, e.g., product development, are conceptualized and built. These tutorial report summaries a research paper that uses a fuzzy approach based on an intelligent agent framework to develop modular products. This approach aims to address the research issue:” How can modular design be carried out through intelligent agents to meet a customer’s fuzzy requirements using modules that come from suppliers that are geographically separated and operate on differing computer platforms?”. The proposed methodology was applied to real-world case that involves module-based synthesis at one of the largest distribution centers of the world GFTZA(Global Free Trade Zone Association) [1]

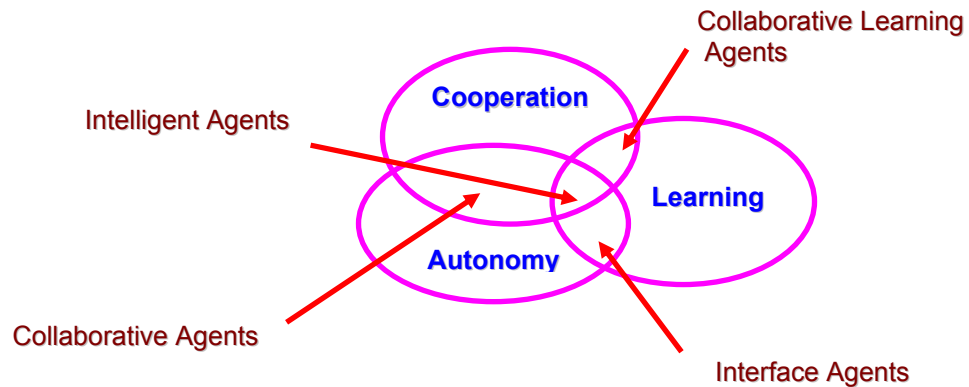
## **1. Introduction**

Business is undergoing a major paradigm shift, moving from traditional management into a world of agile organizations and processes. An agile corporation should be able to rapidly respond to market changes. Corporation managers make informed decisions based upon a combination of judgments and information from marketing, sales, research, development, manufacturing, and finance departments. Ideally, all relevant information should be brought together before a judgment is exercised. However obtaining pertinent, consistent and up-to-date information across a large company is a complex and time consuming process. For this reason, corporations have been seeking to develop a number of information technology (IT) systems to assist with various aspects of the management of their business processes. Such systems aim to improve the way that information is gathered, managed, distributed, and presented to people in key business functions and operations.

The notion of an intelligent agent (IA) is one of the most important concepts to emerge in IT systems in the 1990s; as Guifoyle [2] noted, “ in 10 years time most new IT development will be affected, and many business processes will contain embedded intelligent agent-based systems”

IAs are viewed as “software components and/or hardware which are capable of acting exactly to accomplish tasks on behalf of its user and learn as they react and/or interact with their external environment [1] [2].” The partial view of an agent typology is presented in Fig 1. There are three main components: cooperation, autonomy, and learning.

**Fig.1. Partial View of an Agent Typology**



Agent technology looks set to radically alter not only the way in which computers are interacted but the way complex systems are conceptualized and built as well. The choice of agents as a solution technology was motivated by the following observations [3] [2].

- The domain involves an inherent distribution of data, problem solving capabilities, and responsibilities.
- The integrity of the existing organizational structure and the autonomy of its subparts must be maintained.
- Interactions are fairly sophisticated, including negotiations, information sharing, and coordination.
- The problem solution cannot be prescribed entirely from start to finish.

When taken together, this set of requirements leaves agents as the strongest solution candidate.

- Distributed object systems have the necessary encapsulation, but not the sophisticated reasoning required for social interaction or proactive behavior.
- Distributed processing systems deal with the distributed aspect of the domain but not with the autonomous nature of the components.

Product development is one of the business processes in which this set of requirements must be satisfied. The problems involved in the product development

- are often of such size and complexity that no single individual, organization, or design environment is capable of effectively addressing all aspects of the design;
- include “limited policies for recording experience” in acquiring and using system knowledge within companies;
- “Exhibit a disinclination to search for solutions” which is significant due to the scarcity of experts and support staff for documentation retrieval, etc., which relates to the knowledge access problem category.

What is also becoming clear is that the further requirements of high variety and rapid product development are gradually being superimposed upon these older requirements, so that, for example:” The complex product markets of the 21<sup>st</sup> century will demand the ability to quickly and globally deliver a high variety of customized products [4].” In such situations, the product development problem, specifically the synthesis problem, must be addressed by a team of specialists or IAs. Traditional approaches to product development are based upon a working hypothesis in which it is possible to synthesize hierarchically structured systems by reasoning with models representing components. An underlying problem using the traditional approaches is the softness in many of the probability estimates for specific fuzzy requirements. The real-world data sets from customers who do not have expertise in engineering design are frequently represented with fuzzy features. Such fuzzy synthesis decisions are one of the most conceptually difficult faced by managers. The difficulty arises because, by their nature, such decision must come to grips with uncertainties surrounding highly unlikely requirements with major potentially adverse impacts upon the design process. Many computer-aided design and engineering environments have been developed, but few are capable of supporting the type of distributed, multidisciplinary.

One promising way to address this difficult set of requirements is to use a *fuzzy* approach with embedded agent-based systems, where products are built using standard modules with the incorporation of IAs. *Modules* refer to components whose functional, spatial, and other interface characteristics, fall within the range of variations allowed by the specified standardized interfaces of a modular product design. The mixing and matching of modules in modular product design can potentially generate a large number of different products, each consisting of distinct combinations of components that give each product distinctive functions, features, and/or performance levels.

Producing modular products by combining modules in consideration of product performance is becoming the focus and frequently stated design goal of agile manufacturing, which enables corporations to respond rapidly to changes in the market environment.

The research approach summarized in this report promises the benefits of high volume production (that arises from producing standard modules) while, at the same time, being able to produce a high variety of products that are customized for individual customers.

The research approach aims to address the resulting research issue:

“How can module design be carried out to meet a customer’s fuzzy requirements using modules that come from suppliers that are geographically separated and operate on differing computer platform?”.

The remaining of this report describes the work undertaken in this research approach to conceptualize modular product development through a collection of IAs. The IA framework of the research approach is presented in section two, Section three describes the fuzzy approach applied. Section four illustrates the system implementation with an example. Section five gives an overview and evaluation of the research paper and section six concludes the report.

## **2. Research Approach: Use of IA Framework**

An IA framework called an Intelligent Module-Based Synthesis (IM BS) agent system was designed to enhance the existing process of product development rather than to modify it [5], although the process may be modified before such a system is built. The IMBS researches both the technology and the methods that are needed to improve the way fuzzy information is gathered, managed, distributed and utilized in key business functions and operations. The system characteristics include the following.

- *Intelligence*: The agent automatically customizes itself to the preferences of its customer (client), based on previous experience and imprecise information from interaction with customers. The agent also automatically adapts to changes in its environment.
- *Autonomy*: An agent is able to take the initiative and exercise a nontrivial degree of control over its own actions through service agreements.
- *Cooperation*: An agent does not blindly obey commands but makes suggestions to modify requests or ask clarification questions. It also cooperates with other agents to query the modules needed.

The tasks of the agents involve

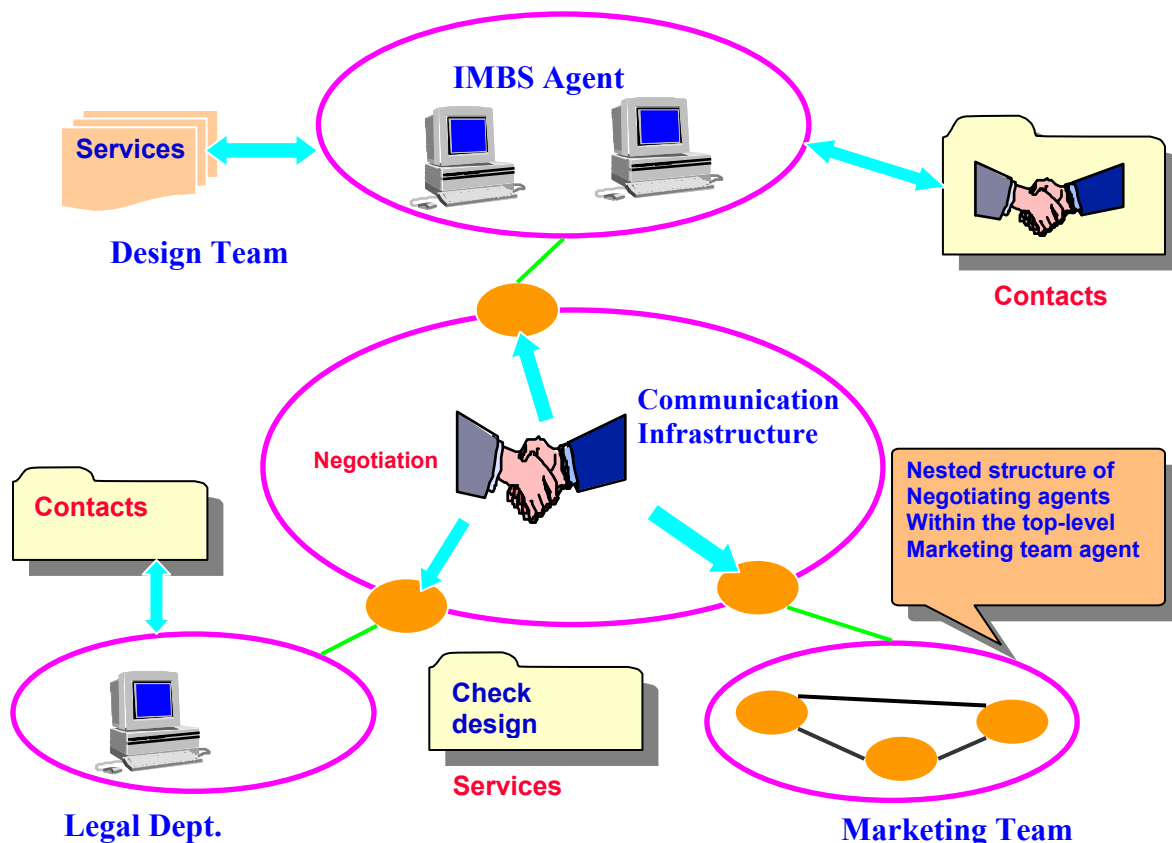
- selecting modules to satisfy the fuzzy requirements from customers;
- synthesizing the modules into an integrated product;
- Coordinating and scheduling the processes intelligently.

### **2.1. Intelligent Module-Based Synthesis agent System.**

In the IMBS agent system, each IMBS agent is able to perform one or more services (Fig. 2). A service corresponds to some problem solving activities of product synthesis. The simplest service (called a *module synthesis job*) represents a problem solving atomic activity endeavor in the IMBS agent system, e.g., combining two modules into a higher level module. These atomic activities can be combined to form complex *services*, e.g., creation of a personal computer (PC) including a terminal, a motherboard, keyboard modules, etc., by adding ordering constraints (e.g., two tasks can run in parallel, must run in parallel, or must run in sequence). The nesting of services can be arbitrarily complex and at the topmost level the entire business process ultimately can be viewed as a service.

Service requirements are issued either from other department, e.g., market teams through an intranet, or from external customers through the Internet. Services are associated with one or more agents that are responsible for the management and execution of those services. Each service is managed by one agent, although the execution of its sub services may involve a number of other agents. Since agents are autonomous, there are no control dependencies between them. Therefore, if an agent requires a service, which is managed by another agent, it cannot simply instruct that agent to start the service. Rather, the agents must come to mutually acceptable agreement about the terms and conditions under which the desired service will be performed. This agreement is like a “recipe” made up of two types of primitives: jobs that are executed entirely within the control of the agent; services provided by other agents over the infrastructure.

**Fig.2. Environment of IMBS Agent System**

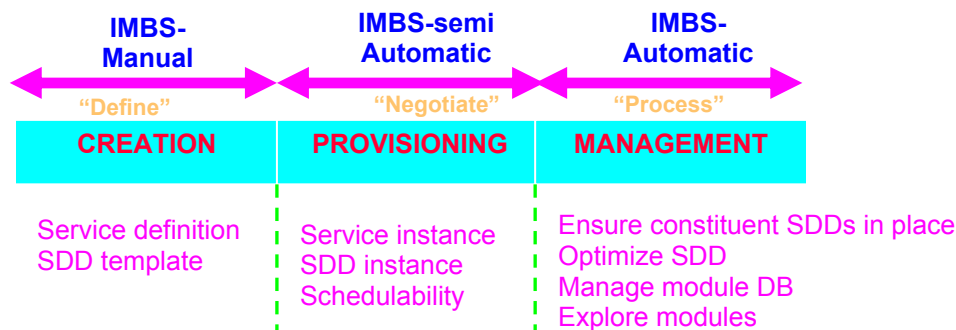


The mechanism for making agreements is *negotiation*- a joint decision making process in which the parties verbalize their (possibly contradictory) demands and then move toward agreement by a process of concession or search for new alternatives.

To negotiate with one another, agents need a *protocol* that specifies the role of the current message interchange, e.g., whether the agent is providing a service, responding with a counter-solution, or accepting or rejecting a service. Additionally, agents need a means of

describing and referring to the domain terms involved in the negotiation. For example, both agents need to ensure that they are describing the same service even though they may both have a different (local) name for it and represent it in a different manner. This heterogeneity is inherent in most organizations because each department typically models its own information and resources in its own way. Thus when agents interact, a number of semantic mapping and transformations may need to be performed to create a mutually comprehensive semantic map that can be used as an *information sharing language* [8].

**Fig.3. Service Lifecycle**



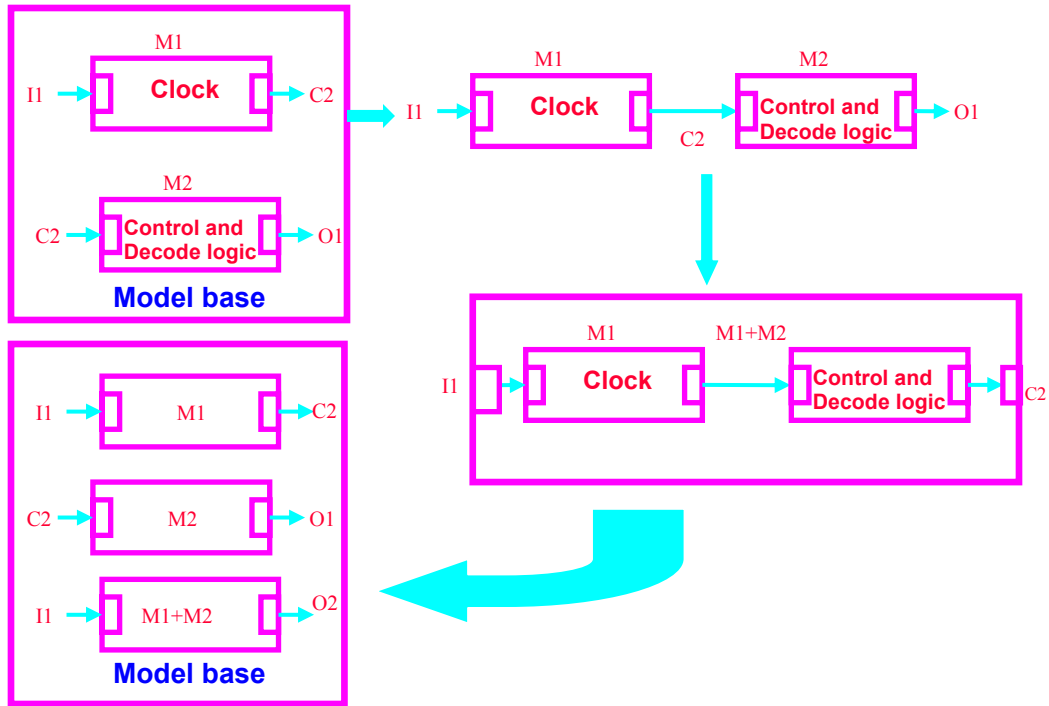
There are three distinct phases to the service lifecycle (Fig.3). First, the agent has to describe the service and how it is realized. This is carried out using an IMBS *Service Description Diagram* (SDD) template. A SDD template is described by a name, input, and output fields of the modules and combined module. The name uniquely identifies the service provided by that agent.

The input and output field specifies

- What module information is needed by the service;
- Who is to provide or receive it;
- If it is mandatory (must be provided before the service can start) or optional (if available it will be used, but if it is unavailable the service can still proceed).

As an illustration, Fig.4 shows a service description from a customer quote business process. The service synthesizes modules *M1* and *M2* into module *M1+M2* (called module synthesis operation that maps and add desired modules to the current model).

**Fig.4. Illustrative SDD template**

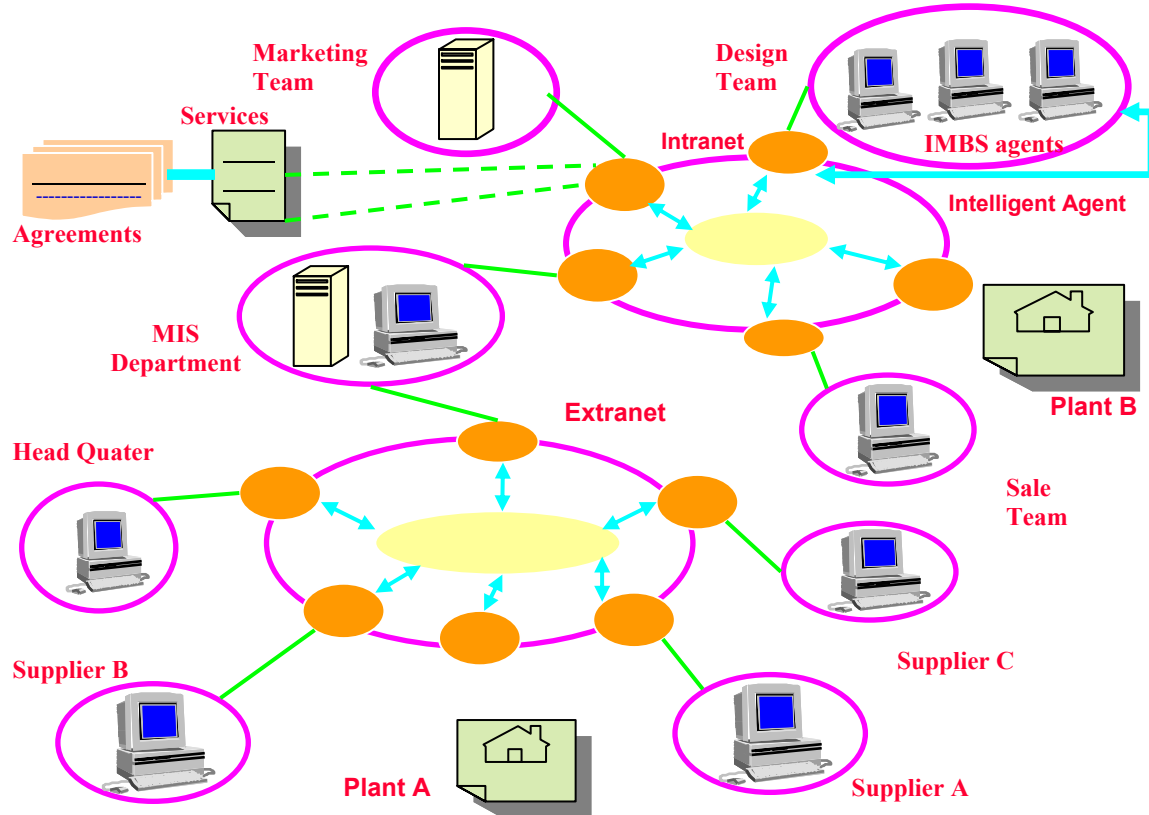


The service is named “ $M1 + M2$ ” and has both of its inputs available: input  $I1$  from the client agent and input  $C2$  from the server agent. The output field specifies the information produced by the service (in this case it is  $M1 + M2$  module and the output  $O2$ ). In the example shown, the service consists of three sub services  $(J1, J2, J3) = (\text{job\_explore\_}M1, \text{job\_explore\_}M2, \text{and job\_add}(M1, M2))$  which must be executed in parallel ( $J1$  and  $J2$ ) and then sequenced (With  $J3$ ). Associated with  $ok_{M1+M2}$  is a completion expression, which specifies that each of the sub services must successfully be completed in the entire service is to succeed.

Second, in the provision phase the feasibility that the service will be scheduled is verified and the SDD template specification is confirmed.

In the management phase, the service is provided. Because the service is executed by an autonomous IMBS agent in an unpredictable environment, it is felt that fuzzy control scheduling decisions are best left for the agent to determine during the run-time. Thus, in this system, the service specifies a partial control flow with some restrictions on the order and degree of concurrency in the execution.

The overview of the IMBS agent system is presented in Fig.5.

**Fig.5. Overview of IMBS Agent Environment**

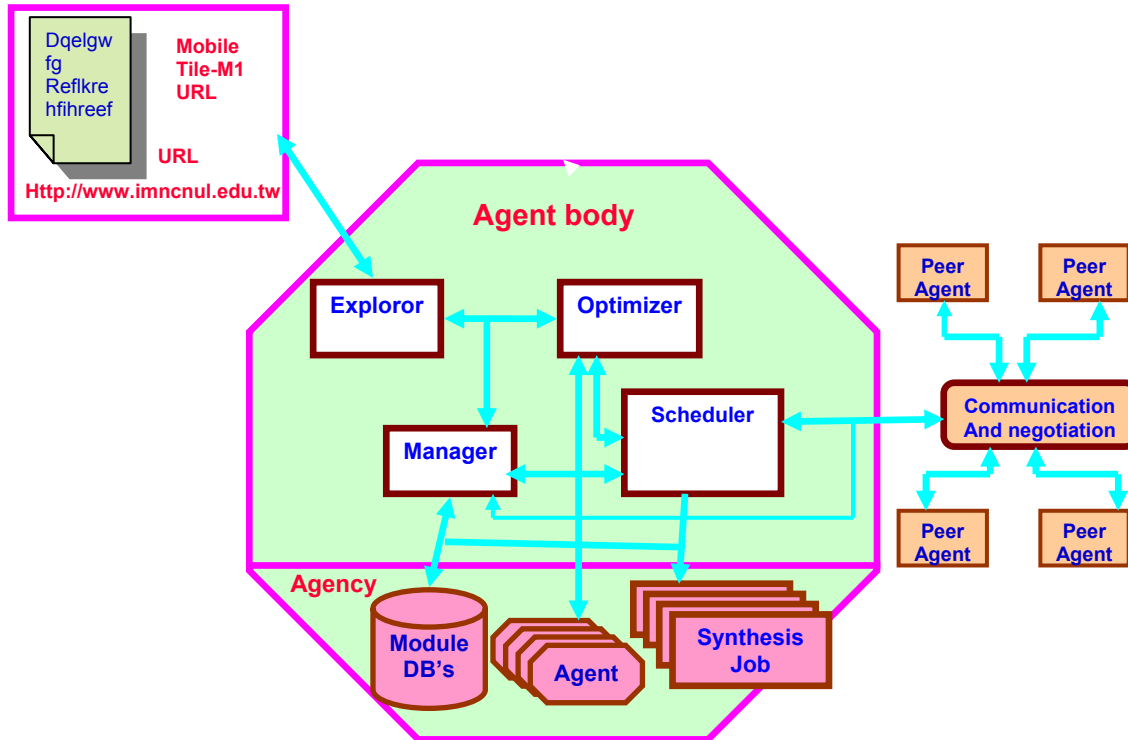
## 2.2. Architecture of an IMBS Agent

The IMBS agents play four roles in the modular product development.

- A *scheduler* is an agent that schedule the module synthesis operations
- A *manager* is an agent that communicates with other agent and maintains the module database where module information is stored.
- An *optimizer* is an agent that optimizes the module synthesis based upon the requirements from customers and engineering constraints.
- An *explorer* is an agent that searches the modules that are located in other distributed databases.

All IMBS agents have the same basic architecture (Fig.6).

This involves an *agent body* that is responsible for managing the agent's activities and interacting with peers and an *agency* that represents the solution resources for product synthesis problems. The body has a number of functional components responsible for each of its main activities-scheduling module synthesis operations, searching desired modules, optimizing module synthesis, and managing the module databases.

**Fig. 6. IBMS architecture** (adapted from ADEPT, [6])

The domain resources include not only module databases and module synthesis jobs but other IMBS agents as well. The latter case allows a nested (hierarchical) agent system to be constructed in which higher level agents realize their functionality through lower level agents (the lower level agents have the same structure as the higher level agents and, can therefore, have subagents as well as module synthesis jobs in their agency). For example, the higher level agent may represent a product design department whose is carried out by a number of design teams (the lower level agents). This structure enables flat, hierarchical, and hybrid organizations to be modeled in a single framework.

The differences between an agent in an agency and a peer agent relate to the levels of autonomy and helpfulness. In both cases, the agents negotiate to reach agreements: however, in the former case we have the following.

- The Agent cannot reject the proposal outright (although it can counter propose until an acceptable agreement is reached).
- The agent must negotiate in a cooperative (rather than a competitive) manner (since there is some degree of commonality of purpose).

In summary, there is a tight coupling between an agent and its agency, and between an agent and its peers in two ways:

- loosely coupled and b) tightly coupled. In a loosely coupled interaction each agent has an equal status, no one agent controls another. In tightly coupled mode, one agent is a controlling agent and the other agents have access restricted from agents outside of the agency.

*Scheduler:* Responsible for assessing and monitoring the agent ability to meet i) the SDD agreement that is already agreed upon and ii) the potential SDD agreement that it may agree to in the future.

This involves two main roles: scheduling and exception handling. The former involves maintaining a record of the availability of the agent's resources, which can then be used to determine whether SDDs can be met or new SDDs can be accepted. The exception handler receives exception reports from the optimizer during service execution (e.g., "service may fail" "service has failed," or "no SDD in place") and decides upon the appropriate response. For example, if a service is delayed then the scheduler may decide to locally reschedule it, to renegotiate its SDD, or to terminate it altogether.

*Optimizer:* Responsible for optimizing the service results through the execution. Three main roles involves: service execution management (optimizing executed services as specified by the agent's SDDs) solution presentation (routing solutions between servers, clients and other agents) and exception handling (monitor the execution of jobs and services for unexpected events and then react appropriately).

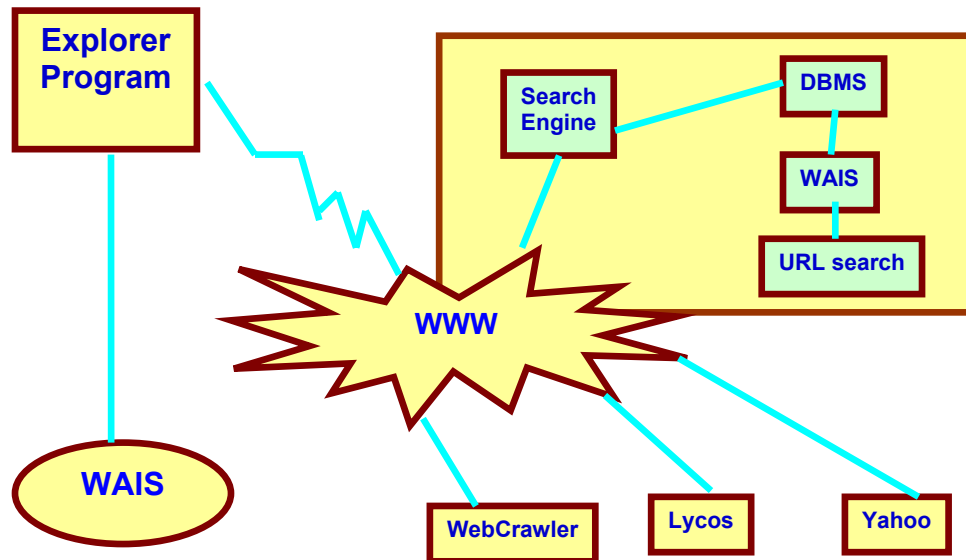
*Manager:*

- 1) Maintains and provides access to 1) the SDDs agreed upon with other agents and a list of peers which can provide services of interest and 2) module database.
- 2) Delivers the status messages of active services.
  - a) Between the optimizer and the clients.
  - b) Between an agent and its agency.
    - Between the optimizer and agents within the agency during service execution.
    - Between the scheduler and agents within the agency during negotiation.
  - c) Between peers agents
    - Between the optimizer and peer agents during service execution
    - Between the scheduler and peer agents during negotiation.
- 3) Communicates between the optimizer and clients within the agency relating to job management activities (e.g., activate, suspend, or resume a job).
- 4) Communicates between agents within that agency or peer agents relating to service execution management (e.g., an instruction to start service, service finished, service results). The scheduler's communication with both agency agents and peer agents relates to service negotiation.

*Explorer:* perform the role of managing, querying or collating modules information from many distributed sources. It is able to module information from many distributed sources. It is able to traverse the WWW, gather information and report what it retrieves to a home location. Fig.7 shows how the typical static explorer works. It indicates how an explorer is associated with some particular search engines. A search engines is able to search the WWW, depth-first, and store the topology of the WWW in the data base management system (DBMS) and the full index of URLs in the WAIS [1]. Public search/indexing engines such as Lycos or Webcrawler can be used similarly to build up the index. The explorer, which has been requested to collected information on some

modules, issues various search requests to one or several URL search engines. The information is collated and sent back to the IMBS agent.

**Fig.7. View of how an explorer works**



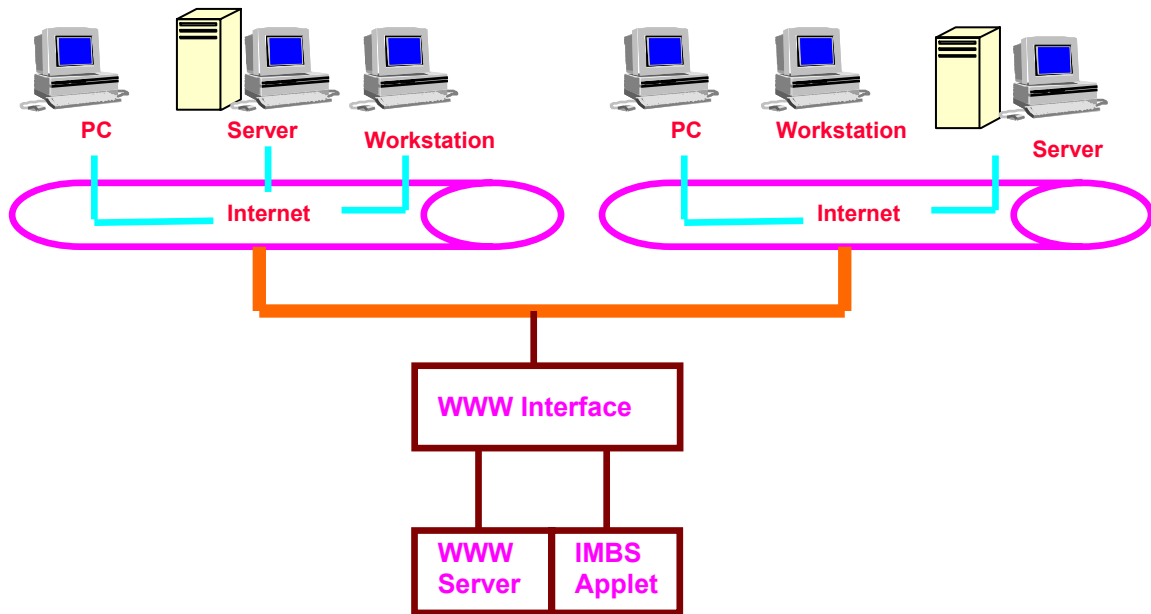
### 3. Fuzzy Approach

A fuzzy approach was used in the agent body that solves the issues of scheduling modules synthesis operations, monitoring that status of active services and optimizing the module synthesis. The fuzzy approach was implemented with the java language in IMBS applets. Decision makers or clients use the browsers to input the requirements and run the IMBS applets through the World Wide Web (WWW) regardless of what platform are used (Fig.8). The WWW is potentially useful for remote decision making since it allows the disparate functions that are involved in remote decision making to share data relatively easily.

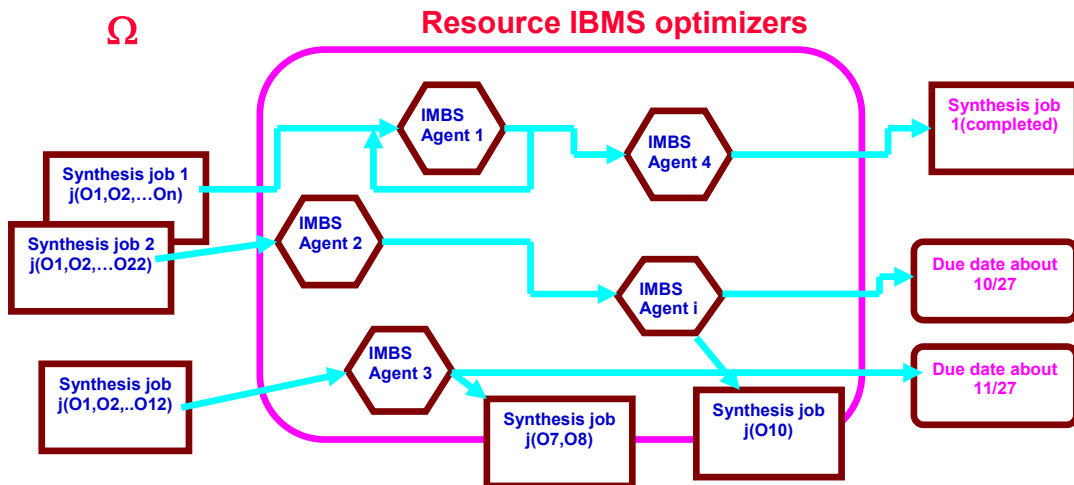
In the traditional research literature of business process management, most models are based upon the assumption that the problem domain does not contain any imprecision, vagueness, subjectivity, or precise information. However, this assumption is strongly challenged in the implementation phase of such models by the fact that imprecision is inherent in almost all problem domains unless the system under consideration is totally mechanized, closed, and predictable, which seem to be valid only in the laboratory environments used for prototyping. Modeling an imprecise problem domain using precise modeling tools not only results in unpredictable behavior in the model but total failure in many cases because of the misrepresentation of the knowledge content.

In Section 3.1, a fuzzy scheduling approach is presented to schedule module synthesis operations. Each module synthesis operation involves creating product variants derived from different combinations of existing or new modules using a fuzzy module-based synthesis approach that is presented in Section 3.2

**Fig.8. Overall Architecture of a Fuzzy Approach through the WWW**



**Fig.9. Scenario of Scheduling Activities**



### 3.1 Fuzzy scheduling approach:

A typical scheduling problem can be described as follows: a set of jobs (module synthesis jobs) must be performed by means of a set of resources (IMBS optimizers). Each job  $j$  requires the scheduling of a set  $\Omega_j$  of operations (module synthesis operations) according to a process plan (SDD) that satisfies precedence constraints. A scenario of scheduling activities is presented in Fig.9., where module synthesis job 1 is completed, and module synthesis jobs 2 and  $j$  are processed by IMBS agents  $i$  and 3 with operations  $O7$ ,  $O8$ , and  $O10$  remaining, respectively. Once started, operations cannot be interrupted. In the simplest situation, each operation  $O_i$  must be performed by a given resource and has a precise duration  $t_i$ . Capacity constraints between two operations requiring the same resource express that these cannot overlap in times. Let  $s_i$  denote the starting time of  $O_i$  that the scheduling procedure must compute. Propagating the release dates and the due dates of job  $j$  over operation  $O_i$  must take place;  $r_i$  is its release date (earliest starting time) and  $d_i$  its due date (latest ending time).

#### *Fuzzy Constrained Scheduling Problem:*

a) Flexible Temporal Constraints: In agile business processes, release and due dates of module synthesis jobs are often subject to preference. For instance module synthesis job  $j$  must absolutely be completed at the latest completion date, and it should preferably be completed before the due date, or as soon as possible after this due date, similarly, it is better to start module synthesis job  $j$  after its preferred release date, while it is impossible to start it before the earliest acceptable release date. The requirement about the release date associated to module synthesis job  $j$  is no longer crisp but can be modeled by fuzzy numbers with respect to due date).

Namely, the membership function release date constraint is increasing when the release date goes from min to max. The membership function due date constraint is decreasing when due date goes from min to max. Taking into account the preference constraints, temporal constraints attached to each module synthesis job lead to define a fuzzy temporal window for each module synthesis operation  $O_i$ .

b) Feasible Schedules: A solution to a fuzzy scheduling problem is typically an assignment of starting times of every module synthesis operation. It must satisfy precedence constraints, capacity constraints, release, and due date constraints. While release and due dates are flexible and, capacity and precedence constraints remain crisp. An assignment satisfying precedence and capacity constraints satisfies the fuzzy schedule problem insofar as it satisfies the least satisfied temporal constraint. The global satisfaction level depends on the chosen starting times for modules synthesis operations.

#### *Solving Fuzzy Constrained Scheduling problems:*

The solving approach consists in searching for a sequencing of module synthesis operations by IMBS schedules, from which earliest and latest starting times can be computed Disjunctive constraints (e.g.,  $O_i$  before  $O_k$  or  $O_k$  before  $O_i$ ) must be transformed into simple precedence constraints by the resolution precedence, which is based on three basic procedures.

- A consistency enforcing procedure which propagates the effects of the scheduling decisions updating the precedence (conjunctive) graph.

- A general search procedure that precedes by managing new precedence constraints.
- A constraint analysis procedure (or look-a head heuristic) that determines the precedence decision to make next.

A linear algorithm LP has been implemented that computes the fuzzy temporal windows according to the precedence constraints.[1]

This method guarantees that the best among the earliest starting times according to the precedence constraints can then be obtained when assigning to each  $s_i$  the lowest date (greatest date with respective to latest starting times) among its best possible values.

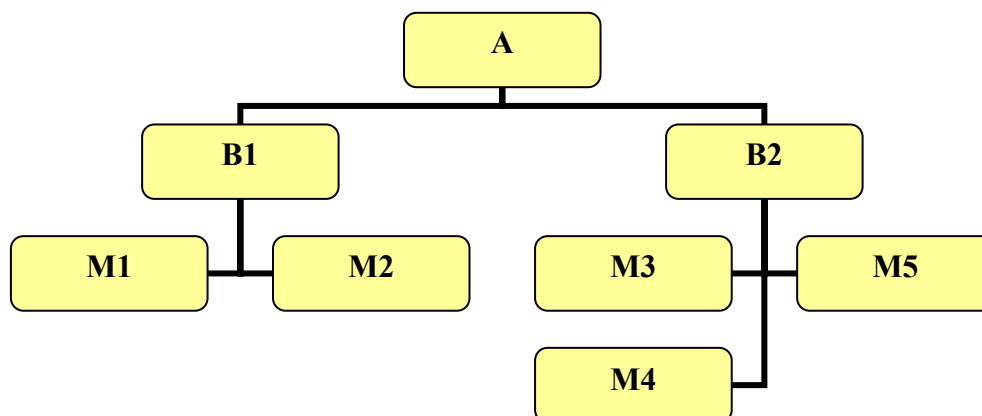
### 3.2 Fuzzy Module-Based Synthesis Approach

The synthesis in engineering design is concerned with a systematic generation of alternatives in order to create new designs or improve the existing ones. Model synthesis in engineering design has gain a considerable attention. The research explores the following:

*Module Hierarchy:* Most decision makers confronted by complex decision problems often try to decompose a major system into subsystems. There are, however, some cases in which they could not clearly represent the system itself or not suitably decompose the major system. The approach developed in the research paper allows the decision maker to describe the system in a vague or flexible way. An illustrative module hierarchy is shown in Fig.10. Five modules,  $M1$ ,  $M2$ ,  $M3$ ,  $M4$ , and  $M5$ , make up product A. These aggregation functions,  $f_1$ ,  $f_2$ , and  $f_3$ , show a customer's preference structure.

*Fuzzy Evaluation of alternatives based Upon the Customer Preference:* Interaction procedure: alternatives are evaluated on the attainment levels of the lowest-level objectives  $M_i$  ( $i = 1, \dots, 5$ ) from zero to one by customers. Since it cannot be expected that a customer evaluate the five modules precisely and simultaneously, the hierarchical evaluation as shown in Fig.10 can be employed. These aggregation functions show a customer's preference structure. The approach requires the customer to fill in a preference form throw the WWW to identify each  $f$ . In the reference form, the desired modules are selected and their weights are chosen.

**Fig. 10** Illustrative Example of Module Hierarchy

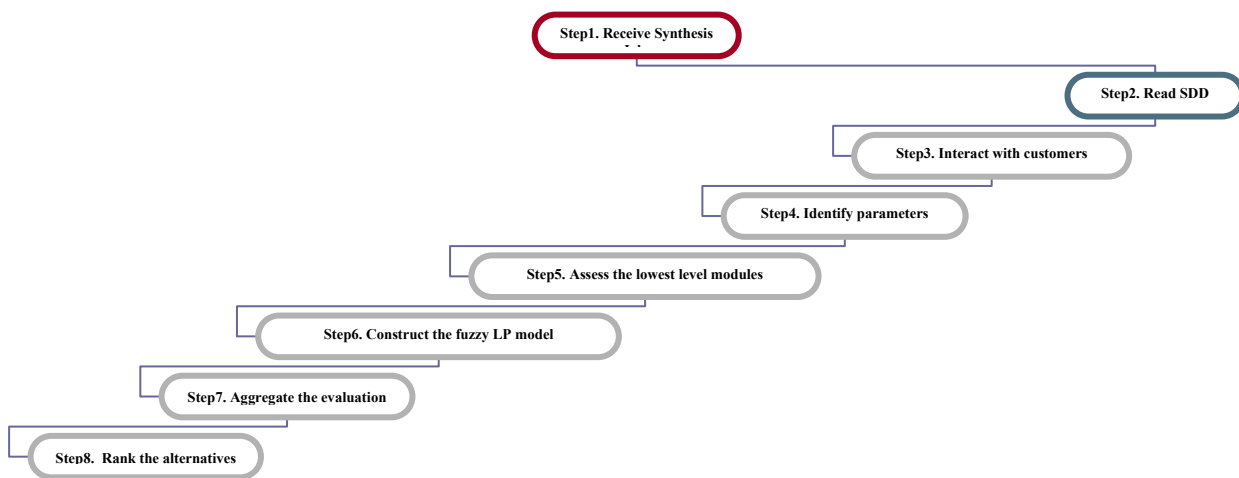


*Fuzzy evaluation:* After the operation is determined, the optimizer assesses the lowest-level objectives for all the alternatives by the previous scale values. However, there is branching since the objectives cannot be assessed uniquely. Fuzzy probability representation is employed in the approach for this situation.

Finally, the fuzzy expected evaluation is defuzzified with its centroid. The results are to support the decision making of module synthesis.

We can notice that the Fuzzy Module synthesis Procedure is useful for imprecise information and could be summarized using the following eight steps (Fig.11).

**Fig. 11: Fuzzy Module Synthesis Procedure**



The optimizer

1. receives a module synthesis job from the scheduler;
2. reads the SDD of the module synthesis job and constructs its module hierarchy;
3. interacts with the customers to identify a preference model for the module hierarchy;
4. identifies the parameters of the preference model represented by three fuzzy operations:  $\lambda$  operation and its two extensions, and selects one desired operation among the three as the optimum preference model [1];
5. assesses the customer's evaluations on the lowest-level module for alternatives: if a unique values cannot be assessed, fuzzy probability is applied;
6. Constructs the fuzzy LP(Linear Programming) testability models and solves the models.
7. aggregates the evaluation on the lowest –level modules and evaluates the solution of fuzzy LP testability modules;
8. ranks the alternatives

#### **4 System Implementation:**

The research approach presented in the paper [1] was applied to a real-world case which involves module-based synthesis at one of the largest distributed centers in the world, the Global Free Trade Zone Association (GFTZA), Inc. The customer inputs the request for a new service. The scheduler calculates a desired completion date and suggests to the client. The client may choose “Not accept” to start the negotiation with that IMBS agent, or simply “accept” it. An example will be, when for instance four days exceeds the expected due date in the previous input. After that, the optimizer presents the module hierarchy, interacts with the client to fill in his/her preference for each module, and sets up the desired total cost in a fuzzy way. Based on the engineering constraints, and customer preferences, the optimizer outputs three alternatives. Now, the client can choose one as the final alternative, or go back the previous step to adjust his/her preference for each module until the final decision is made.

*Customer Feedback:* This approach has been used at GFTA, Inc, since august 1997. After each service, the feedback is collected and measured with four indices.

- % reduction of service redo (from 46% in previous systems to 11% with the current system).
- % increasing of customer’s satisfaction (from 45% in previous systems to 73% with the current system).
- % reduction of customer complaints (from 34% in previous systems to 18% with the current system).
- % reduction of services overdue (from 32% in previous systems to 12% with the current system).

#### **5 Overview and Evaluation of the Research Approach**

This research paper presented an IMBS agent system that used a fuzzy approach based upon an intelligent agent framework to develop modular products. The IMBS agent system was applied to a real world case that involved module-based synthesis at one of the largest distribution centers in the world. This system challenged two critical difficulties faced in business process.

- Imprecision inherent in almost all problem domains while the business processes under consideration are not totally mechanized, closed and predictable.
- Aim of information technology systems, e.g., agent technology at improving the way information is gathered managed, distributed, and presented to people in key business functions and processes.

The following issues require further investigation

- Expansion to other business processes other than product development.

- Extension and more rigorous specifications used in negotiation: There are few evaluative studies of negotiation, and most of these focus on the effects of different negotiation strategies upon the agent society.

## **6 Conclusion**

1. In the course of this tutorial report I have been able to learn about a real application case study of Intelligent Agents System, and found out how using intelligent agents to manage fuzzy businesses. The topic covers the basic part of the material given in the course SENG609-22 such as:
  - Agent topology and components.
  - Methodology for agent oriented analysis and design, where the key concepts are:
    - i. Domain characteristic.
    - ii. Conceptual frame work.
    - iii. System, roles, responsibilities, protocol, and architecture.
2. This topic addresses a strong need to agent systems development methods in managing fuzzy businesses and supporting decision making. In the traditional research literature of business process management [7], most models are based upon the assumption that the problem domain does not contain any imprecision, vagueness, subjectivity, or precise information. However, this assumption is strongly challenged in the implementation phase of such models by the fact that imprecision is inherent in almost all problem domains. And modeling an imprecise problem domain using precise modeling tools not only results in unpredictable behavior in the model but total failure in many cases because of the misrepresentation of the knowledge content. So this research proposed a fuzzy approach to deal with these issues.
3. The framework used a distributed multi agents system that relies on knowledge bases built with little sharing and reuse in the business processes of product development only.
4. This research topic shows how future intelligent system development needs libraries of ontology, rather than building from scratch, the approach was applied for product development , so the available of such libraries could save much time and effort for different business processes.
5. An expansion of this approach to other domains will greatly decrease the development time while improving the robustness and the reliability of the future resulting knowledge based in business development.

## 7 References

1. Chun-Che Huang, "Using Intelligent Agents to Manage Fuzzy Business Processes", *IEEE 2001*.
2. SENG 609-22, Dr Far course notes, Department of Electrical and Computer Engineering, university of Calgary.
3. N. Jennings and M. Wooldridge, "Software agents," *IEE Rev.*, vol.42, pp.17-20, 1996
4. W. Davidow and M. Malone, *The virtual Corporation*. New York: Harper Collins, 1992.
5. J.L Alty et al., "ADEPT-Advanced decision environment for process tasks: Overview and architecture," in *Proc. BCS Expert Syst. Conf.(Applicat. Track)* Cambridge, UK., 1994, pp.359-371.
6. N.R. Jennings et al., "using intelligent to manage business processes" in *Proc. First Int. Conf.Practical Applicat. Intell. Agent Multi-Agent Technol.*, London, UK., 1996, pp.345-360.
7. S.W. Sanderson and V. Uzumeri, " Strategies for new product development and renewal: Decision-based incrementalism," *Cent. Sci. Technol. Policy*, Rensselaer Polytech. Inst., Troy, N.Y, Working Paper, 1990

## 8 Acknowledgement

I would like to thanks Dr Far for putting together the material of the course SENG609-22 in the way that's capture the topic and open the door for the students to explore the area of software engineering agent base.