Web Intelligence and Agent Systems: An international journal 5 (2007) 255–271
IOS Press

Building agent-based hybrid intelligent systems: A case study

Zili Zhang[a,b,*] and Chengqi Zhang[c]
[a]*Faculty of Computer and Information Science, Southwest University, Chongqing 400715, China*
[b]*School of Engineering and Information Technology, Deakin University, Geelong Victoria 3217, Australia*
[c]*Faculty of Information Technology, University of Technology, Sydney, PO Box 123 Broadway, NSW 2007 Australia*

Abstract. Many complex problems (e.g., financial investment planning, foreign exchange trading, data mining from large/multiple databases) require hybrid intelligent systems that integrate many intelligent techniques (e.g., fuzzy logic, neural networks, and genetic algorithms). However, hybrid intelligent systems are difficult to develop because they have a large number of parts or components that have many interactions. On the other hand, agents offer a new and often more appropriate route to the development of complex systems, especially in open and dynamic environments. Thus, this paper discusses the development of an agent-based hybrid intelligent system for financial investment planning, in which a great number of heterogeneous computing techniques/packages are easily integrated into a unifying agent framework. This shows that agent technology can indeed facilitate the development of hybrid intelligent systems.

Keywords: Multi-agent systems, intelligent agents, agent-based systems, hybrid intelligent systems, financial investment planning

1. Introduction

Many complex problems such as financial investment planning and foreign exchange trading, data mining from large/multiple databases involve different components or sub-tasks, each of which requires different types of processing. To solve such complex problems, a great diversity of intelligent techniques including traditional hard computing techniques (e.g., expert systems) and soft computing techniques (e.g., fuzzy logic, neural networks, and genetic algorithms) are needed. For example, in the financial investment planning application [31], neural networks can be used as a pattern watcher for the stock market [13, pp. 85–88]; genetic algorithms can be used to predict interest rates [20]; the approximate reasoning based on fuzzy logic can be used to evaluate clients' financial risk tolerance ability [2]. These techniques (called *intelligent techniques* in this paper) are complementary rather than competitive and thus must be used in combination and not exclusively [28]. As a result, such systems must be *hybrid intelligent* ones [9].

However, it is difficult to apply conventional software development techniques for developing such hybrid intelligent systems.

A typical development cycle in the implementation of hybrid intelligent systems was given in [9]. There are six stages in the construction of hybrid intelligent systems: problem analysis, property matching, hybrid category selection, implementation, validation, and maintenance. Most current hybrid intelligent systems are built either from scratch or by following this development process. There are some shortcomings of the hybrid intelligent systems by following this development process. The most obvious one is that the organization of such a hybrid system is not adaptive. Once the techniques are selected in the property matching stage, it is difficult to change or replace it even though one may find a better one later on. Another difficulty lies in the hybrid category selection phase. At hybrid category selection phase, the developers must choose the type of hybrid system required for solving the specific problem. This is not an easy job to do. The hybrid

*Corresponding author. E-mail: zhangzl@swu.edu.cn.

intelligent systems' inherent complexity means it is impossible to know *a priori* about all potential links or relationships among components consisting of a system; interactions will occur at unpredictable times, for unpredictable reasons, between unpredictable components. For this reason, it is futile to try and predict or analyse all the possibilities at design time.

Fortunately, some researchers in agent research community have given a qualitative analysis to provide the intellectual justification of precisely why agent-based methodology is well suited to engineering complex software systems [12]. Now it is generally accepted in agent community that agents offer a new and often more appropriate route to the development of complex systems, especially in open and dynamic environments [16]. On the other hand, hybrid intelligent systems are complex software systems because they have a large number of parts or components that have many interactions. Thus a multi-agent perspective is suitable for modelling, designing, and constructing hybrid intelligent systems [30]. Furthermore, the flexible nature of agent interactions means that agents can make decisions about the nature and scope of interactions at run-time rather than design time. This can overcome the shortcomings mentioned above.

In this paper we demonstrate how agent technology can facilitate the construction of hybrid intelligent systems. In particular we discuss the analysis, design and implementation of an agent-based hybrid intelligent system for financial investment planning. Each intelligent technique has particular strengths and weaknesses, and they cannot be applied universally to all situations of financial investment planning problems. Furthermore, a collection of agents are needed for complex decision making during the course of such a planning. Hence, it is necessary to integrate two or more intelligent techniques with multiple agents. On the other hand, as different techniques can be easily integrated into one hybrid intelligent system under a unifying agent framework, many complex problems can be solved in a shorter time frame. Also due to a variety of complementary problem solving techniques/approaches are combined together, higher-quality solutions can be produced with such systems.

There has been some work that also involved in this topic. One of such attempts is the MIX multi-agent platform [10]. Another such attempt is the PREDICTOR system [9, pp. 153–173]. Khosla and Dillon [15] introduce a computational architecture called IMAHDA (Intelligent Multi-Agent Hybrid Distributed Architecture). A more recent attempt is the multi-agent architecture for fuzzy modelling [6]. Delgado et al. [6] proposed a hybrid learning system that combines different fuzzy modelling techniques by means of a multi-agent architecture. Jacobsen [11] proposed a generic architecture for hybrid intelligent systems.

Among the above agent-based hybrid framework or systems, the MIX, PREDICTOR, and the architecture for fuzzy modelling only integrated very limited soft computing techniques. Both the MIX and PREDICTOR systems are focused on the integration of neural networks and symbolic technologies such as expert systems. The multi-agent architecture of Delgado et al. concentrated on the integration of different fuzzy modelling techniques such as fuzzy clustering, fuzzy rule generation, and fuzzy rule tuning techniques. In MIX and PREDICTOR systems, the way for integrating intelligent techniques into multi-agent systems is to embed the intelligent techniques in each individual agent. The MIX and IMAHDA architectures are not flexible enough as no middle agent [5] was used. The work in [11] is focused on the micro (intra-agent) level of agents, i.e., the integration and interaction of different components within one agent. The macro (inter-agent) level integration and interaction are ignored.

By conducting some experiments and comparisons, the agent-based hybrid intelligent system described in this paper has the following crucial characteristics that differentiate this research from others such as the works in [6], [9, pp. 153–173], [10,11,15]:

- Any new capabilities (in the form of additional agents) can easily be added to the system, and any techniques no longer used can be deleted from the system dynamically at run-time.
- The flexibility and robustness of such systems are greatly improved.

The remainder of this paper is structured as follows. Section 2 is the description of the investment planning problem and the technical models used to solve the problem. The architecture for the agent-based investment planning system is discussed in Section 3. The analysis and design of the system based on Gaia are detailed in Section 4. Decision aggregation is important for the performance of such a system. Section 5 discusses how to aggregate the decisions from different agents. The implementation details are described in Section 6. The experimental results and evaluation are also presented in this section. Finally, Section 7 concludes this paper.

2. Description of the investment planning problem

When a person wants to invest some money somewhere, he usually turns to the financial investment adviser for advice. The first thing the adviser needs to do is to understand the client's individual circumstances. The adviser may ask the client to provide the following information about himself: his financial position (annual income, total net-worth etc.), age, tax effectiveness, and his investment attitude (aggressive or conservative) etc. Based on the information, the adviser should evaluate the client's financial risk tolerance ability and adjust it according to the interest rate trend (falling or increasing). With the client's risk tolerance as well as his age information, the adviser then provides advice to the client on how to allocate portions of his investment across the three main asset types: savings, income, and growth (asset allocation). This should be based on some investment advisory model.

Suppose the adviser suggests the client to invest the growth part of his investment in the stock market after evaluating his financial risk tolerance ability. How can one select a portfolio for the client under his constraint (risk tolerance level, return rate etc.)? The adviser should gather some information about the stock market. The information includes market data, financial report data, technical models, analysts' reports, breaking news etc. After gathering the information, the adviser then makes a portfolio selection decision based on some models (e.g., the Markowitz model, the fuzzy probability model, etc.). This is a typical scenario for investment planning.

3. The architecture for the agent-based investment planning system

In order to identify which components should be contained in a typical financial planning system, without loss of generality, consider a financial house providing investment advice for clients. In such a house, there are: a front counter or reception desk clerk, one or more personnel officer(s), and many financial investment experts (decision makers). The advice giving (decision making) process is initiated by a user contacting the front desk clerk with a set of requirements. The clerk asks the personnel officer to provide the experts' profile, and then delegates the task to one or more experts based on experts' profiles. The experts then work on the task and try to give their recommendations with or without external help. After the experts finish preparing a recommendation (if the task was assigned to more than one expert, the recommendations from different experts must be combined to form a final one), they pass it to the front desk clerk. Finally, the clerk sends the advice to the user. Such a typical process can help us to identify the agent types in a multi-agent system for financial planning.

Based on the typical scenario and process for financial investment planning, seven types of agents are required in such an agent-based system. These are user agents, interface agents, planning agents, middle agents, service provider agents, decision making agents, and decision aggregation agents. The behaviors of each kind of agent in the system are briefly described below:

Interface Agent. This agent interacts with the user (or user agent). It asks the user to provide his personal information and requirements, and provides the user with a final decision or advice that best meets the user's requirements.

Planning Agent. The planning agent is in charge of the activation and synchronization of different agents. It elaborates a work plan and is in charge of ensuring that such a work plan is fulfilled. It receives the assignments from the interface agent.

Decision Making Agent. It is application-specific, i.e., it has its own knowledge base; it must have some meta-knowledge about when it needs the help of intelligent technique agents (e.g., pre or post processing some data); it can ask intelligent technique agents to accomplish some sub-tasks.

Middle Agent or Serving Agent. The serving agent is a matchmaker – one kind of middle agent [5]. It keeps track of the names, ontologies, and capabilities of all registered intelligent technique agents in the system; it can reply to the query of a decision making agent with appropriate intelligent technique agent's name and ontology.

Service Provider Agent. Most of the service provider agents in the system are intelligent technique agents (the agents based on different *intelligent techniques* are called *intelligent technique agents*). Each intelligent technique agent can provide services for decision making agents with one or some kind of combined intelligent techniques; it can send back the processed results to decision making agents; it must advertise its capabilities to the serving agent.

Decision Aggregation Agent. When decision making agents finish the assigned tasks they return the results to the decision aggregation agent. The aggregation agent chooses one of the alternative decisions, or

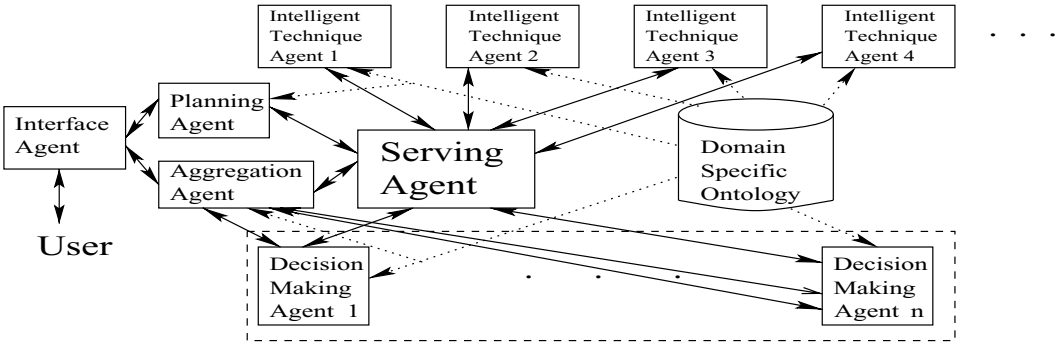


Fig. 1. Architecture of Agent-Based Hybrid Intelligent System.

performs an aggregation of the different results into a final one.

The architecture of the financial planning system is shown in Fig. 1. Such an architecture can ensure the flexibility, robustness, and interoperability of the resulting system. In particular, this architecture can provide supports for the following: Any agent in the system can access any of the intelligent techniques available in the system when needed.

The ontology indicated in Fig. 1 is the foundation for agent communication. All agents in the system interpret the content of received messages based on the ontology. To avoid packing too much information into one paper, the ontology related issues are not discussed in this paper. We will provide details for the behaviors of decision aggregation agents and the algorithm used for aggregation in the system in Section 5. The empirical evaluation is given in Section 6.2.

4. Analysis and design of the system

This section describes how to determine the roles and functionality of different agents systematically. Existing software development techniques are inadequate for modeling agent-based systems. Existing approaches fail to adequately capture an agent's flexible, autonomous problem-solving behavior, the richness of an agent's interactions, and the complexity of an agent system's organizational structures. For these reasons, an agent-oriented methodology is required to analyze and design such an agent-based hybrid intelligent system.

4.1. A brief introduction to Gaia methodology

There are dozens of agent-oriented methodologies available. How can we choose the appropriate one for our task? First of all, the main characteristics of the agent-based hybrid intelligent system for financial investment planning can be identified as follows:

- Agents are heterogeneous, in that different agents may be implemented using different programming languages, architectures, and techniques;
- The organizational structure of the system is dynamic, in which agents can dynamically leave and enter the system;
- Agents exhibit social behavior, in that they interact with one another to cooperate to achieve a common objective;
- There are no self-interested agents in the systems; and
- Integration and interaction of different techniques is crucial.

After comparing a few agent-oriented methodologies, *Gaia* [21] was chosen. In Gaia, analysis and design are well-separated phases. Analysis aims to develop an understanding of the system and its structure, in terms of the roles that have to be played in the agent organization and of their interactions, without any reference to implementation details. The design phase aims to define the actual structure of the agent system, in terms of the agent classes and instances composing the system, of the services to be provided by each agent, and of the acquaintances' structure. Based on Gaia, the emphasis of the analysis and design is to identify the key *roles* in the system and document the various *agent types* that will be used in the system.

The role model identifies the key roles in the system. Here a role can be viewed as an abstract description of an entity's expected function. A role is defined by four attributes: *responsibilities*, *permissions*, *activities*, and *protocols*. The relationships of the four attributes can be described as follows, which is based on [21] and [22, pp. 289–290].

Role Schema:	*name of role*
Description	*short English description of the role*
Protocols and Activities	*protocols and activities in which the role plays a part*
Permissions	*"rights" associated with the role*
Responsibilities	
Liveness	*liveness responsibilities*
Safety	*safety responsibilities*

Fig. 2. Template for Role Schemata.

Responsibilities are the key attribute associated with a role as the functionality of a role is determined by them. There are two types of responsibilities: *liveness properties* and *safety properties*. Liveness properties are used to describe those states of affairs that an agent must bring about under certain given environmental conditions. Safety properties are *invariants*, i.e., that an acceptable state of affairs is maintained across all states of execution. Intuitively, a liveness property states that "something good happens", while a safety property states that "nothing bad happens".

A role has a set of *permissions* or "rights" to implement responsibilities. The permissions of a role thus identify the *information resources* that are available to that role in order to realize its responsibilities. For example, a role might have associated with it the ability to read a particular item of information, or to modify another piece of information. A role can also have the ability to generate information. The *activities* of a role are computations associated with the role that may be carried out by the agent without interacting with other agents.

Finally, a role is also identified with a number of *protocols*, which define the way that it can interact with other roles. A role model is comprised of a set of *role schemata*, one for each role in the system. A role schema draws together the various attributes discussed above into a single place (see Fig. 2).

The formal notation for expressing protocols, activities, permissions, and responsibilities adopted by Gaia will be used. To introduce these concepts, the example of a PRICEWATCHER role will be used. The purpose of this role is to monitor whether the trading price of a specific security is exceeding the expected value of the share holder. The protocols and activities involved in the PRICEWATCHER role include: InformShareholder, GetInitializeInformation, GetPrice, and Compare. The activity names (like Compare) are underlined to distinguish them from protocols.

The following is an illustration of the permissions associated with the role PRICEWATCHER:

`reads` **supplied** *SecurityCode* // Security code used in Share Exchanger
supplied *ExpectedValue* // The value the shareholder expected
supplied *TradingPrice* // The current trading price of the security

This specification defines three permissions for PRICEWATCHER: it says that the agent carrying out the role has permissions to access the value of $SecurityCode$, $ExpectedValue$, and $TradingPrice$. The **supplied** keyword here is used to indicate that some roles are parameterized by certain values. Another two types of permissions are `changes` (read and modify) and `generates` (produce a resource). Note that these permissions relate to the *knowledge* that the agent has.

The liveness responsibilities for the PRICEWATCHER role might be:

– whenever the share exchange is not closed, get the trading price of the specific security (indicated by the $SecurityCode$);
– whenever the trading price is exceeding the expected value, inform the share holder.

Following the Gaia notation, liveness properties are specified via a *liveness expression*, which defines the "life-cycle" of the role and is a regular expression. The general form of a liveness expression is:

ROLENAME = *expression*

where ROLENAME is the name of the role whose liveness properties are being defined, and *expression* is the liveness expression defining the liveness properties of ROLENAME. The atomic components of a liveness expression are either *activities* or *protocols*. The operators for liveness expressions are shown in Table 1.

Role Schema: PRICEWATCHER
Description: This role involves monitoring whether the trading price of a specific security is exceeded the expected value of the share holder.
Protocols and Activities: InformShareholder, GetInitializeInformation, GetPrice, Compare
Permissions: reads **supplied** *SecurityCode* // Security code used in Share Exchanger **supplied** *ExpectedValue* // The value the shareholder expected **supplied** *TradingPrice* // The current trading price of the security
Responsibilities Liveness: PRICEWATCHER = (GetInitializeInformation)+.(GetPrice, Compare)+.(InformShareholder)* Safety: • True

Fig. 3. Schema for Role PRICEWATCHER.

Table 1
Operators for Liveness Expressions

Operator	Interpretation
$x.y$	x followed by y
$x\|y$	x or y occurs
$x*$	x occurs 0 or more times
$x+$	x occurs 1 or more times
x^{ω}	x occurs infinitely often
$[x]$	x is optional
$x\|\|y$	x and y interleaved

Thus the liveness responsibilities of the PRICEWATCHER role can be expressed as:

$$\begin{aligned}\text{PRICEWATCHER} = &(\text{GetInitializeInformation})\\ &+.(\text{GetPrice, Compare})\\ &+.(\text{InformShareholder})*.\end{aligned}$$

This expression says that PRICEWATCHER consists of executing the protocol GetInitializeInformation, followed by the protocol GetPrice, followed by the activity Compare and the protocol InformShareholder.

Safety requirements are specified by means of a list of predicates. These predicates are typically expressed over the variables listed in a role's permission attribute. By convention, safety expressions are listed as a bulleted list, each item in the list expressing an individual safety responsibility.

When all these are put together, the schema for the PRICEWATCHER role results (Fig. 3).

4.2. Gaia-based analysis and design

Again, based on the description of the typical scenario and process for financial investment planning and the Gaia methodology, it is comparatively straightforward to identify the roles in the hybrid intelligent system for financial investment planning. The up-front administrator's behavior falls into two distinct roles: one acting as an interface to the user (USERHANDLER) and one overseeing the process inside the organization (WORKPLANNER). The personnel officer's behavior falls into another two roles: one keeping track of the profiles (CAPABILITYRECORDER) and one checking the profiles (CAPABILITYMATCHER). The experts' behaviors are covered by DECISIONMAKER, HELPPROVIDER, and DECISIONAGGREGATOR roles. The final role is that of the USER who requires the decision. For demonstration purpose, the schema for DECISIONMAKER role is shown in Fig. 4.

With the respective role definitions in place, the next stage is to define the associated interaction models for these agent roles. Here we focus on the interactions associated with the DECISIONMAKER role.

This role interacts with the WORKPLANNER role to obtain the task this role will accomplish (ReceiveTask protocol, Fig. 5a). It interacts with the INFOGATHER role to gather some relevant information (known facts) for the task (GetInformation protocol, Fig. 5b). It also interacts with the CAPABILITYMATCHER role to provide some roles for data pre- and/or post-processing and so on when accomplishing the task (AskforHelp protocol, Fig. 5c). When the DECISIONMAKER role finishes making decision for the task, it informs the DECISIONAGGREGATOR role of its alternative decision for the task (InformDecisionAggregator protocol, Fig. 5d).

Based on the key *roles* identified, the agent model can be generated (Fig. 6). This shows, for most cases, a one-to-one correspondence between roles and agent types. The exception is for the CAPABILITYRECORDER and CAPABILITYMATCHER roles which, because of their

Role Schema: DECISIONMAKER
Description: Reaches a conclusion for the delegated task based on the information gathered and knowledge the role has.
Protocols and Activities: ReceiveTask, GetInformation, AskforHelp, InformDecisionAggregator, Reasoning
Permissions: reads **supplied** *Task* // Task delegated by other roles **supplied** *Knownfacts* // Gathered information generates *Decision* // one alternative decision
Responsibilities Liveness: DECISIONMAKER = ReceiveTask.GetInformation.(Reasoning.AskforHelp)+.InformDecisionAggregator Safety: • True

Fig. 4. Schema for Role DECISIONMAKER.

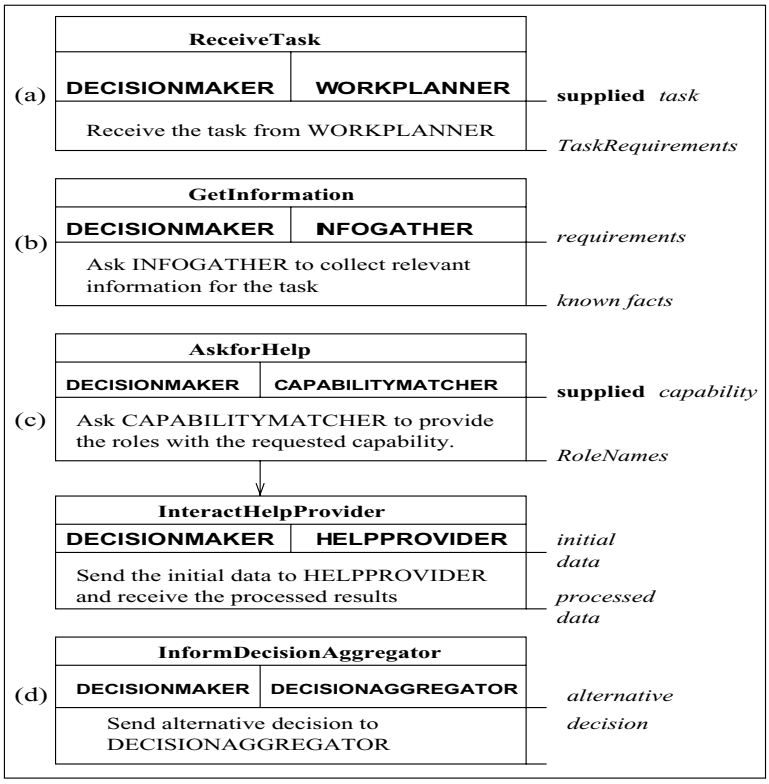


Fig. 5. Definition of Protocols Associated with the DECISIONMAKER Role: (a) ReceiveTask, (b) GetInformation, (c) AskforHelp, and (d) InformDecisionAggregator.

high degree of interdependence, are grouped into a single agent type.

In this system, the most important organisational rule in the organisational model is that *if a role says it has a capability then it can perform the tasks corresponding to the capability and will do so when asked*.

5. Decision aggregation in the system

In the agent-based financial investment planning system, there are agents that have similar problem solving and decision making capabilities. The results from these agents require to be combined. This section discusses how to aggregate different agent's opinions in

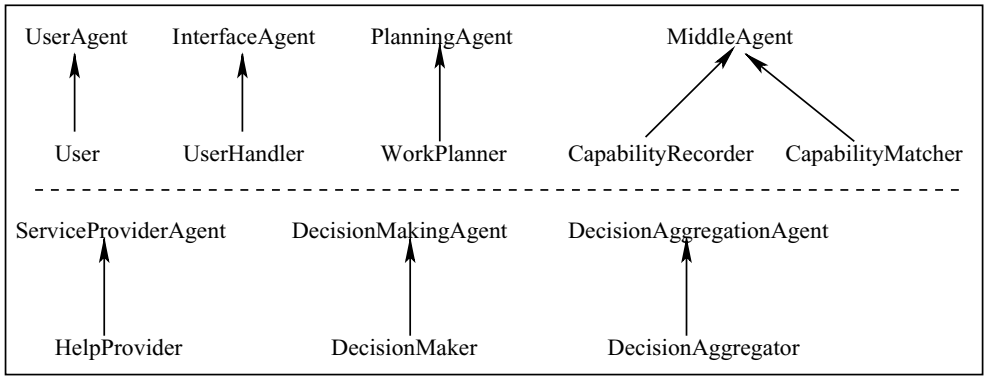


Fig. 6. Agent Model of the System.

general, and the algorithm implemented in the system in particular.

Generally, in multi-agent systems, each of the agents may have its own expertise. When they are asked to make a decision on the same task, the results may be different. In such situations, different decisions need to be aggregated to obtain a final result. For example, suppose a user (investor) wants to know whether his investment policy (IP) should be aggressive or conservative. First, the user gives his *annual income* (AI) and *total net-worth* (TN) to the decision making agents through the interface agent. The decision making agents use their own knowledge (with the help of intelligent technique agents) to evaluate the user's *risk tolerance* (RT) ability using rules such as: *If user's* AI *is low* (L) *and* TN *is* L*, then user's* RT *is* L. Note that different decision making agents may have different rules similar to this.

The decision making agents then delegate the information gathering agents to collect data concerning the rise or fall of interest rates, the state of the stock market, the trade balance, the unemployment rate, the level of inventory stock, and so on. These data are called parameters, and are represented as $P = \{P_1, P_2, \ldots\}$. The parameters collected by different information gathering agents may differ.

Assume there are k parameters to be collected: $P = \{P_1, P_2, \ldots, P_k\}$, and m information gathering agents are asked to collect the k parameters independently. The gathered results are $\{P_{i1}, P_{i2}, \ldots, P_{ik}\}$ $(i = 1, 2, \ldots, m)$. The first aggregation problem involves combining $\{P_{i1}, P_{i2}, \ldots, P_{ik}\}$ $(i = 1, 2, \ldots, m)$ in some reasonable way to obtain $P = \{P_1, P_2, \ldots, P_k\}$.

Now, suppose there are n decision making agents. Each agent has rules in its knowledge base such as

$$\text{If } RT \text{ is } H \text{ and } P_1 \text{ is } B_1 \text{ and } \ldots \quad \text{then } IP \text{ is } C_i \tag{1}$$

where $C_i (i = 1, 2, \ldots, n)$ is a fuzzy subset indicating the aggressive or conservative degree of the investment policy.

Because the knowledge of the decision making agents and their decision attitudes may be different, the answers to the same question may also be different, and differ in various degrees. They have to be combined or reconciled in order to produce a final decision.

There are many aggregation operators such as uninorm operators [25] (the $t-$norm and $t-$conorm [7] are its special cases), general compensation operator (general averaging operator [1] is its special case) and their various weighted/prioritized counterparts (e.g., relative weighted/prioritised t-norm and t-conorm, prioritised compensation operator. Different applications need different aggregation operators. In this financial application, since there exists much fuzzy or uncertain information, the aggregation approaches should be able to deal with such information. Thus, Yager's OWA operator [24,26,27] is chosen in this system. The OWA operator is to provide a family of aggregators having the properties of mean operators and weighted aggregation. Formally, a mapping $F : R^n \longrightarrow R$ is called an OWA operator of dimension n if it has an associated weighting vector W of dimension n such that its components satisfy

(1) $w_j \in [0, 1]$;
(2) $\sum_{j=1}^{n} w_j = 1$; and
(3) $F_w(a_1, a_2, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j$, where b_j is the jth largest of the a_i.

A fundamental feature of this operator is the reordering process which associates the arguments with the weights. This aggregation can be expressed in a vector notation as $F_w(a_1, a_2, \ldots, a_n) = W^T B$. In this expression, W is the OWA weighting vector associated with the aggregation, and B is the ordered argument

vector; where the jth component in B, b_j is the jth largest of the a_i.

Expressing the OWA operator $F_w(a_1, a_2, \ldots, a_n)$ in its vector notation form, $W^T B$ makes very clear the distinct components involved in the performance of this operation. First, there is a weighting vector W; this is required to have components w_j which lie in the unit interval and sum to one. The second part of the OWA aggregation is the vector B, known as the ordered argument vector. This vector is composed of the arguments of the aggregation. To solve a specific problem using the OWA operator, we need to find out the appropriate weighting vector W and the ordered argument vector B.

There are two characterising measures associated with the weighting vector W (see [24,27]). The first of these, the α value of an OWA operator, is defined as

$$\alpha(W) = \frac{1}{n-1} \sum_{j=1}^{n} w_j (n-j) \tag{2}$$

This measure, which takes its values in the unit interval, is determined by the weighting used in the aggregation. The actual semantics associated with α are dependent upon the application in which it is being used. In our case, the α can be the degree that the aggregation prefers decisions with high confidence, or the attitude of the decision making agent.

The second measure is called the dispersion (or entropy) of W and is defined as

$$H(W) = -\sum_{j=1}^{n} w_j \ln(w_j) \tag{3}$$

Equation (3) helps measure the degree to which W takes into account all of the information in the aggregation.

One method of determining these weights, $w_1, \ldots, w_n$, requires the solution of the following mathematical programming problem:

Maximise $-\sum_{j=1}^{n} w_j \ln(w_j)$ subject to

(1) $\alpha(W) = \frac{1}{n-1} \sum_{j=1}^{n} w_j (n-j)$;
(2) $w_j \in [0, 1]$;
(3) $\sum_{j=1}^{n} w_j = 1$.

Assume that the agents' decisions are still represented by trapezoidal numbers. If $C_i = (a_{i1}, b_{i1}, b_{i2}, a_{i2})$, $i = 1, 2, \ldots, n$, are trapezoidal numbers, then

$$\begin{aligned} C_{OWA} = (&F_w(a_{11}, \ldots, a_{n1}), \\ &F_w(b_{11}, \ldots, b_{n1}), \\ &F_w(b_{12}, \ldots, b_{n2}), \\ &F_w(a_{12}, \ldots, a_{n2})) \end{aligned} \tag{4}$$

where F_w is an OWA operator. We now discuss how to decide the weighting vector W and the ordered argument vector B in different situations when aggregating use Eq. (4).

Suppose that the three agents present their decisions on the investment policy by the fuzzy numbers

$$C_1 = (-100, -100, -50, -30),$$

$$C_2 = (-10, 10, 10, 30),$$

$$C_3 = (60, 90, 100, 100)$$

and the weighting vector can be obtained: $W = [w_1, w_2, w_3] = [1/3, 1/3, 1/3]$. The arguments are ordered by their values.

Corresponding to the weighted case, if the arguments are ordered using the values of r_i, i.e., let b_j be the a_i value which has the jth largest of r_i, and let $W = [w_1, w_2, w_3] = [u_3, u_2, u_1] = [0.5, 0.3, 0.2]$. Formula (4) is then used to aggregate. In both cases, the same results are obtained as those using fuzzy averaging.

The problem here is that the degrees of importance in aggregation were not used directly. Actually in this case, the arguments which need to be aggregated are pairs such as

$$(u_1, a_{11}), (u_2, a_{21}), \ldots, (u_n, a_{n1})$$

Here, the formula $G(u, a) = \bar{\alpha}\bar{u} + ua$ is used to transform the tuple into a single value [26, pp. 41–49], where α is defined by Eq. (2). The following are the steps of the procedure:

1. Calculate the α value of the OWA operator:

$$\begin{aligned} \alpha &= \sum_{j=1}^{3} \frac{3-j}{3-1} w_j = w_1 + w_2/2 \\ &= 0.5 + 0.3/2 = 0.65 \end{aligned}$$

2. Transform each of the argument tuples using $G(u_j, a_j) = \bar{\alpha}\bar{u_j} + u_j a_j$, hence

$$\begin{aligned} G(u_1, a_{11}) &= -49.72, G(u_2, a_{21}) \\ &= -2.755, G(u_3, a_{31}) = 30.175 \end{aligned}$$

$$\begin{aligned} G(u_1, b_{11}) &= -49.72, G(u_2, b_{21}) \\ &= 3.245, G(u_3, b_{31}) = 45.175 \end{aligned}$$

$$\begin{aligned} G(u_1, b_{12}) &= -9.72, G(u_2, b_{22}) \\ &= 3.245, G(u_3, b_{32}) = 50.175 \end{aligned}$$

$$\begin{aligned} G(u_1, a_{12}) &= -5.72, G(u_2, a_{22}) \\ &= 9.245, G(u_3, a_{32}) = 50.175 \end{aligned}$$

3. Calculate C_{OWA}

$$\begin{aligned} C_{OWA} = (&F_w(-49.72, -2.755, 30.175), \\ &F_w(-49.72, 3.245, 45.175), \\ &F_w(-9.72, 3.245, 50.175), \\ &F_w(-5.72, 9.245, 50.175)) \\ = &(4.32, 13.62, 24.12, 26.72) \end{aligned}$$

The defuzzification value is 18.87. This still indicates a very cautious investment policy – much more cautious than one not using the degrees of importance.

The concept of agents' decision making attitudes is also important. Because the agents usually have different knowledge, this results in different attitudes when making decisions. Some are aggressive, some conservative. Here, α_i ($\alpha_i \in [0, 1]$) is used to indicate the agents' attitudes. The bigger the value of α_i, the more aggressive the attitude of the decision making agent DA_i.

Suppose there are still three agents, and their attitudes are $\alpha_1 = 0.3, \alpha_2 = 0.5$ and $\alpha_3 = 0.8$. The decisions they make, and their degrees of importance, remain unchanged, as described above.

To aggregate, the first step is to decide the attitude α of all the agents (in this case three). The OWA operator is still used. Degrees of importance are mapped to unit interval as the weighting vector for combining α_i, called $W(\alpha)$, and

$$W(\alpha) = [w(\alpha)_1, w(\alpha)_2, w(\alpha)_3] = [0.5, 0.3, 0.2]$$

Then

$$\begin{aligned} \alpha &= F_{W(\alpha)}(\alpha_1, \alpha_2, \alpha_3) \\ &= \alpha_3 \times w(\alpha)_1 + \alpha_2 \times w(\alpha)_2 + \alpha_1 \times w(\alpha)_3 \\ &= 0.61 \end{aligned}$$

By solving the mathematical programming problem with $\alpha = 0.61$, the weighting vector W is obtained for the final aggregation as follows:

$$W = [w_1, w_2, w_3] = [0.45, 0.32, 0.23]$$

The arguments are ordered according to the values of r_i. The final aggregation using Eq. (4) gives $C_{OWA} = (0.8, 20.7, 36.7, 47.3)$. The defuzzification value according to the *mean of maximum method* in fuzzy averaging is 28.7 [2]. This suggests a policy on the aggressive side of the scale, but a cautious one – more cautious than that using fuzzy averaging. This is because the decision attitude of DA_1 is slightly conservative, but its decision is very conservative. Taking all the information into account, the investment policy should be cautiously aggressive.

If the degrees of importance are used directly in the aggregation in this case, $C_{OWA} = (1.26, 9.93, 21.38, 24.22)$ is obtained. The defuzzification value is 15.66.

6. Implementation and evaluation

Based on the results of analysis and design, we implemented the system. There are two versions. One is under the support of JATLite (Java Agent Template, Lite, *http://java.stanford.edu/*), the other under the support of AgentBuilder (*http://www.agentbuilder.com*). The discussion in this section is based on the JATLite version.

6.1. Implementation

In this system, there are a large number of components that interact in varying and complex ways. This leads to complex behaviour that is difficult to understand, predict and manage. Take one sub-task of financial planning – financial portfolio management – as an example. The task environment has many interesting features, including [18]: (1) the enormous amount of continually changing, and generally unorganised, information available; (2) the variety of kind of information that can and should be brought to bear on the task (market data, financial report data, technical models, analysts' reports, breaking news, etc.); and (3) the many sources of uncertainty and dynamic change in the environment. It is obvious that financial planning is a typical complex problem and hybrid solutions are crucial.

In the implemented agent-based financial investment planning system, the following models/techniques have been integrated together: a client financial risk tolerance model and a client asset allocation model, both are based on fuzzy logic [2]; two interest rate prediction models, one based on neural networks, the other based on fuzzy logic and genetic algorithms [20]; three portfolio selection models – Markowitz's model [17], the fuzzy probability model, and the possibility distribution model [19]; and expert systems with explanation mechanisms. In addition to these models/techniques, an operations research software package called *LINDO* for solving quadratic programming (*http://www.lindo.com/*) and a matrix software package called *MatrixLib* for solving eigenvalues of matrices (*http://www.mathtools.com/*) were also integrated.

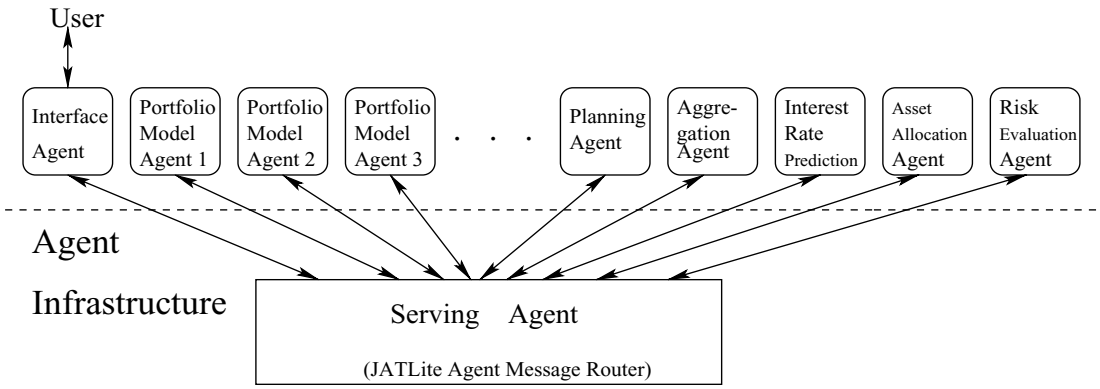


Fig. 7. Practical Architecture of the System.

The architecture of the financial planning system can be derived from the framework directly.

The kernel part of the system is the middle agent. Refer to [31] for implementation details of this middle agent. The introduction of middle agent in the system facilitates the flexibility and robustness. For the details of how and why middle agents can improve the flexibility and robustness of an agent-based system, see [32]. A wrapper was implemented to wrap some legacy software packages and convert them into agents. "To convert legacy programs into agents" means to add a communication layer on top of legacy programs. With the wrapper, all agents can send and receive messages using an agent communication language – KQML (Knowledge Query and Manipulation Language)[14]. Legacy software packages coded in different languages like C/C++ (e.g. interest rate prediction programs), Fortran (e.g. programs solving quadratic programming) are wrapped and easily integrated in the system. Under the support of JATLite, the practical architecture of the system is depicted in Fig. 7.

In the implemented system, the following agents are included: one interface agent (*interFace*), one client's investment policy determination agent (*invpolicy*), two interest rate prediction agents, one based on on neural networks (*ffin*), the other based on fuzzy logic and genetic algorithms (*flga*), one client's financial risk tolerance calculation agent (*invppt*), one stock analysis agent (*stock*), one stock data preparation agent (*stockData*), three portfolio selection agents based on Markowitz's model (*moki*), the fuzzy probability model (*fuzz*), and the possibility distribution model (*poss*), and one decision aggregation agent *aggr*). The relationships of these agents and the high level interactions among these agents are shown in Fig. 8. All agents implemented have the ability to exchange KQML messages. This greatly increases the interoperability of the system.

Figure 9 shows the user interface of the system, which can start from any Internet Browser or appletviewer.

6.2. Empirical evaluation

This subsection discusses the empirical evaluation of the system.

The system can provide reasonable financial investment planning information based on the user provided data and some relevant models. Figure 11 shows the asset allocation results when the annual income is $\$50,000$, networth $\$800,000$, age 35, investment amount $\$30,000$, and investment attitude is aggressive (level 4). By clicking the "explanation" button, the corresponding explanation of how to get the results is displayed in the "result display" window (Fig. 10).

If the growth part is invested in stock market, the system can provide a portfolio for the user (Fig. 11). The portfolio is the aggregated result of three portfolios based on Markowitz's portfolio selection model, the fuzzy probability portfolio selection model, and the possibility distribution portfolio selection model, respectively. The four portfolios are marked as Powa, Pmar, Pfuz, and Ppos, respectively. The aggregation algorithm used is ordered weighted averaging (OWA) aggregation algorithm [29]. By clicking the "evaluation" button, the system will provide the comparisons of the four portfolios (Fig. 12). An empirical evaluation of the aggregated results is given in the following.

At this stage, one important problem is how to verify the aggregated portfolio. There is no systematic way available to answer this question. Instead, some experiments were conducted.

Table 2
Portfolios and Variances Based on 12 Years Return Data

P	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	Variance
P_{POS}	5.15			13.95	19.53			39.75	21.62	0.30
P_{FUZ}					23.05		46.55		30.40	0.04
P_{MAR}					23.14		46.60		30.26	0.05
P_{OWA}	2.85			7.73	21.12		20.77	22.02	25.51	0.18

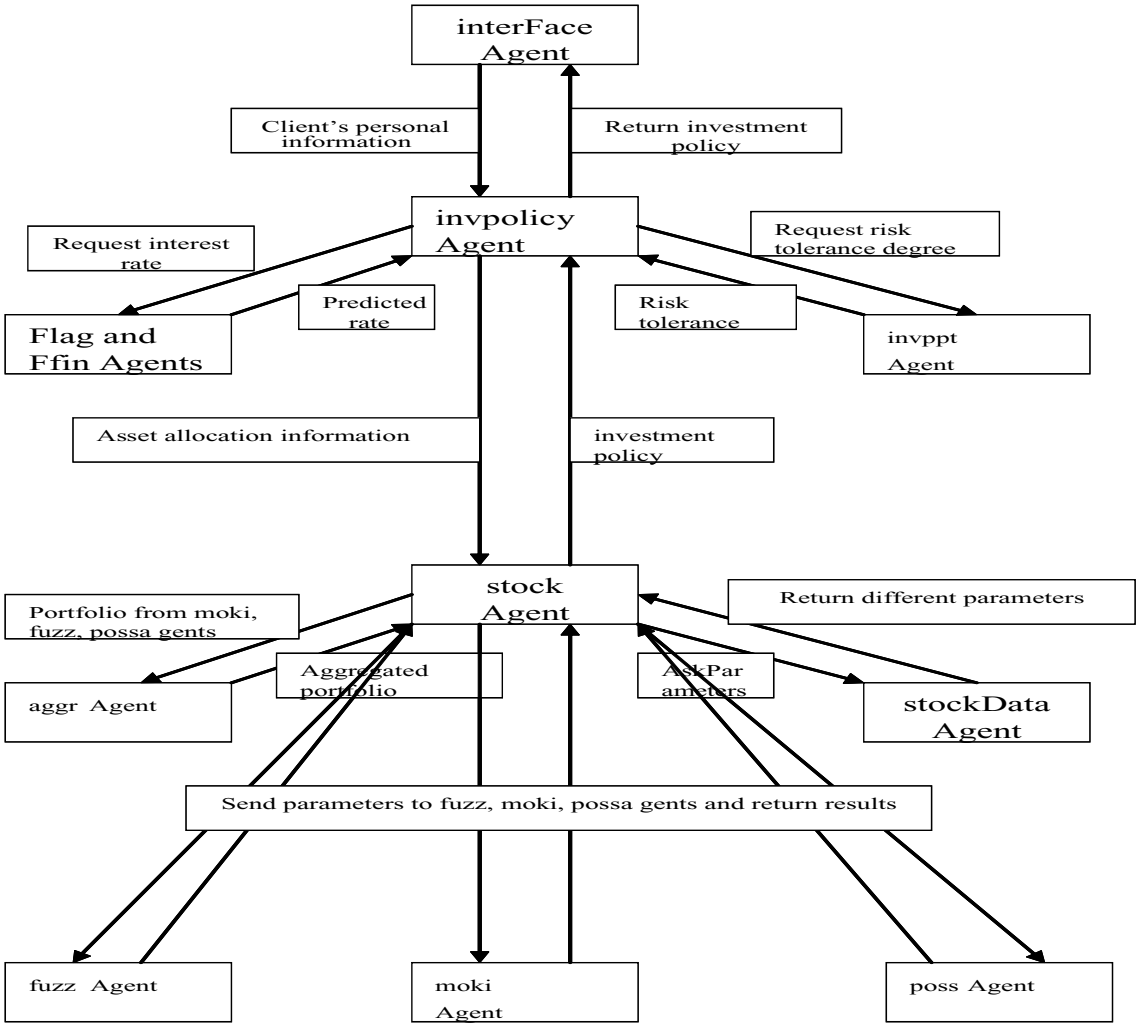


Fig. 8. Relationships among Agents.

The first experiment conducted was to use the first 12 years return data described in [17] and produce three portfolios based on the three models. Based on the analysis in [19], it is known that the fuzzy model is a direct extension of Markowitz's model, while the possibility model is more reasonable than the fuzzy model. Thus the three portfolios are ordered as P_{POS}, P_{FUZ}, and P_{MAR}, and $\alpha = 0.7$ (the degree that the aggregation prefers decisions with high confidence) is chosen when using OWA operator to aggregate the three portfolios. The weight vector with $\alpha = 0.7$ is $W = [0.554, 0.292, 0.154]$. The selected portfolios as well as corresponding risks of investment are shown in Table 2. The portfolios in Table 2 are also selected with an expected average return $r_s = 17\%$.

The last 6 years return data in [17] are used to verify the realized average returns of the four portfolios. The realized average returns of the four portfolios from one

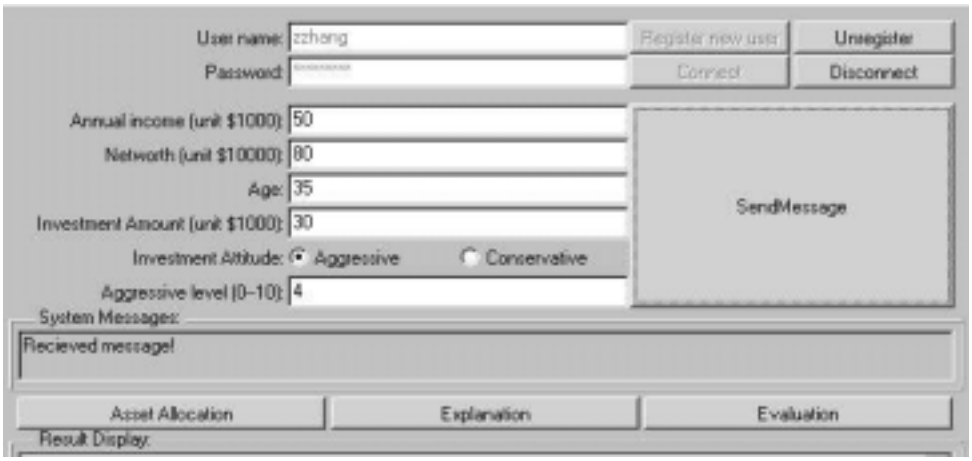


Fig. 9. User Interface of the System.

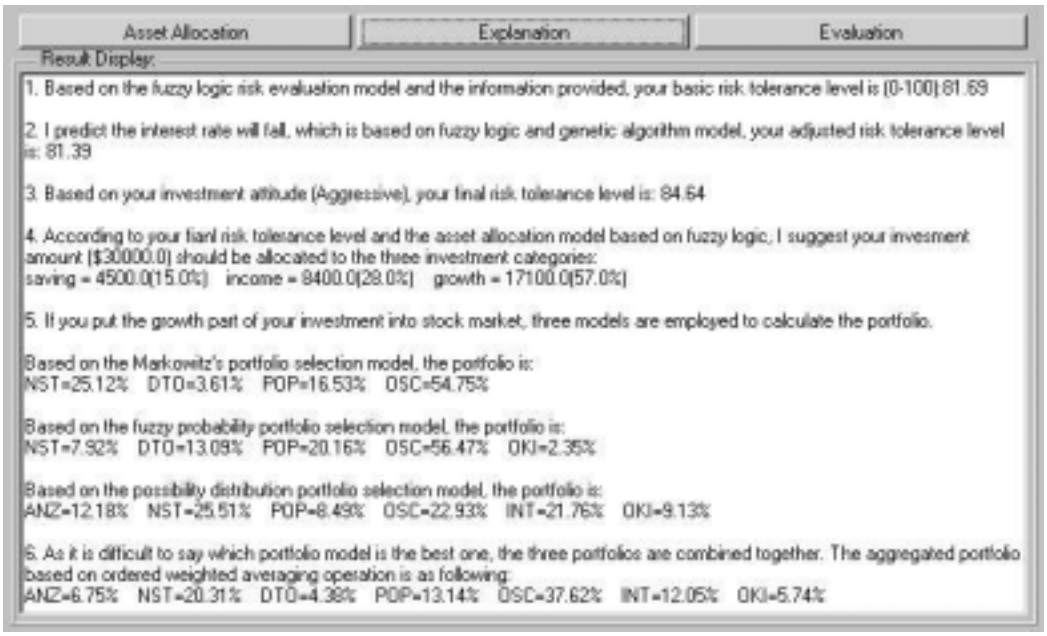


Fig. 10. Example Explanations

year to six years are listed in Table 3.

From Table 3, one can see that the average returns of P_{OWA} are better than those of P_{FUZ} and P_{MAR}, and slightly less than those of P_{POS}. The variance (risk or uncertainty degree of the investment) of P_{OWA} is greatly reduced (from 0.30 to 0.18) compared with that of P_{POS}.

To further verify the aggregated portfolio, 12 securities listed in Australian Stock Exchange Limited (ASX) were selected and 12 years average returns (from 1986 to 1997) were collected (see Table 4). The ASX security codes of S_1 to S_{12} are AKC, AFI, AGL, BPC, CSR, EML, GUD, SMI, HAH, OPS, PDP, and WYL, respectively.

Table 3
Realized Average Returns of the Portfolios (%)

Year(s)	P_{POS}	P_{FUZ}	P_{MAR}	P_{OWA}
1	11.421	7.501	7.554	9.681
2	27.694	15.665	15.700	22.334
3	18.748	14.457	14.467	16.836
4	19.684	14.990	15.006	17.593
5	14.005	13.318	13.334	13.701
6	19.703	14.590	14.607	17.425

Similar to experiment one, the first 8 years (1986 to

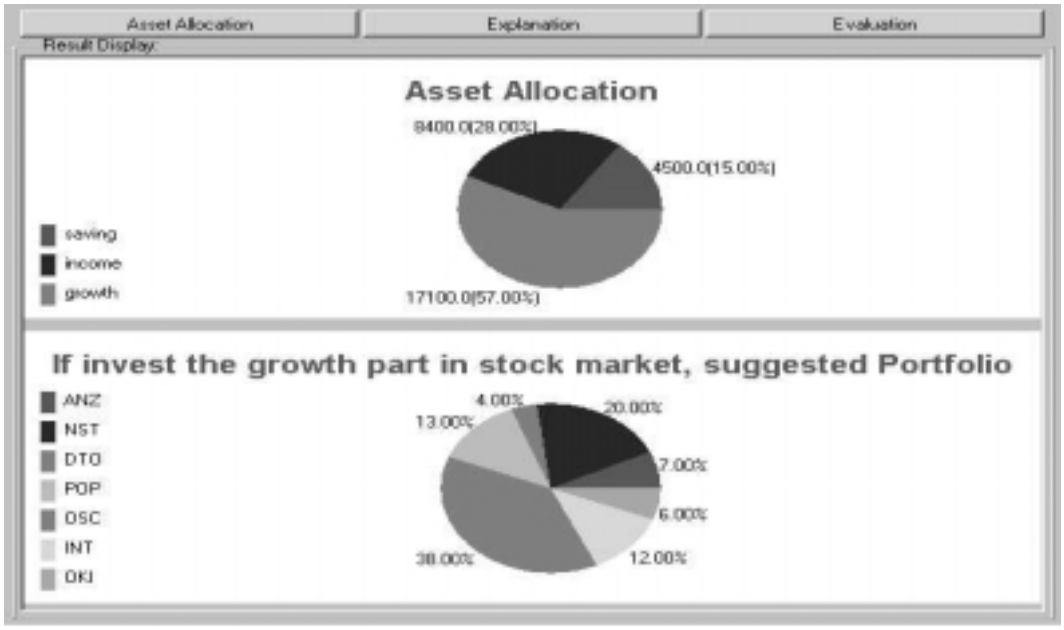
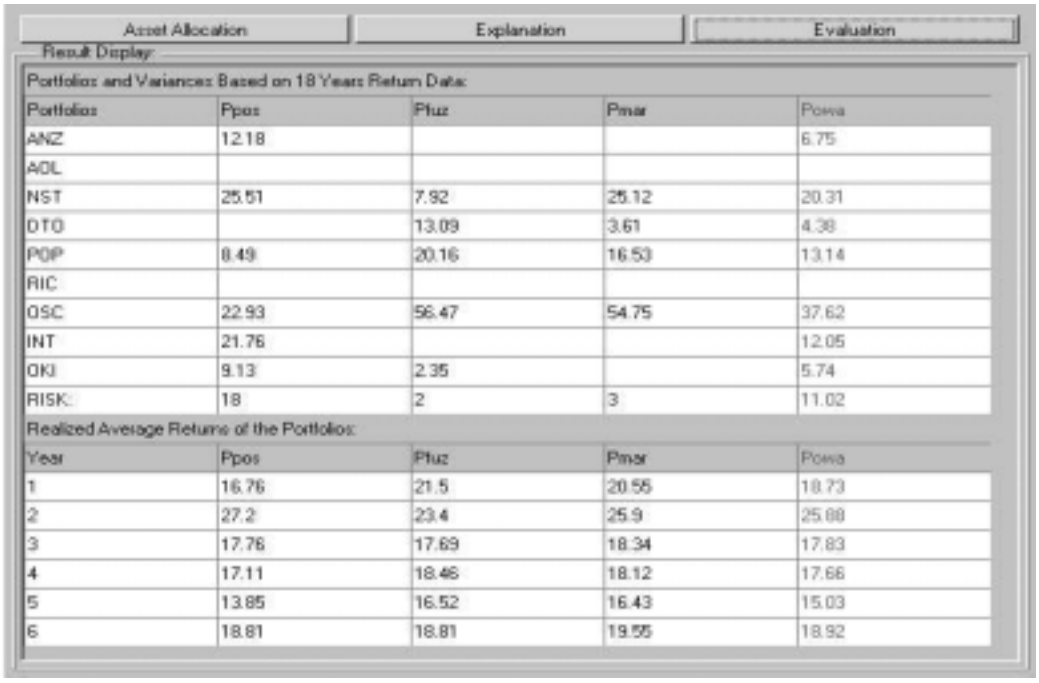


Fig. 11. Example Asset Allocation Results.



Portfolios	Ppos	Pluz	Pmar	Powa
ANZ	12.18			6.75
AOL				
NST	25.51	7.92	25.12	20.31
DTO		13.09	3.61	4.38
POP	8.49	20.16	16.53	13.14
RIC				
OSC	22.93	56.47	54.75	37.62
INT	21.76			12.05
OKI	9.13	2.35		5.74
RISK:	18	2	3	11.02

Realized Average Returns of the Portfolios:

Year	Ppos	Pluz	Pmar	Powa
1	16.76	21.5	20.55	18.73
2	27.2	23.4	25.9	25.88
3	17.76	17.69	18.34	17.83
4	17.11	18.46	18.12	17.66
5	13.85	16.52	16.43	15.03
6	18.81	18.81	19.55	18.92

Fig. 12. Comparison of the Portfolios.

1993) return data was used to generate the portfolios, while the last 4 years (1994 to 1997) data was used to verify. When the expected average return $r_s = 17\%$, the selected portfolios based on the three models and the aggregated portfolio based on OWA with $\alpha = 0.7$ are listed in Table 5. Based on the four portfolios, the realized average returns from one year to four years are shown in Table 6.

The results in Table 6 are consistency with those in Table 3. Thus the same conclusion can be reached. The

Table 4
Returns on Twelve Securities from ASX

Yr	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
86	0.675	0.33	0.053	0.946	0.081	10.026	0.42	0.198	0.405	0.249	0.75	0.08
87	0.648	0.93	0.864	1.097	0.826	0.897	0.535	0.207	1.023	0.333	0.994	0.684
88	0.248	0.432	−0.157	0.656	0.515	1.044	0.623	0.05	0.132	0.174	0.338	0.403
89	−0.007	0.405	0.838	0.698	0.136	0.544	0.358	1.296	0.468	0.271	0.355	0.467
90	−0.087	−0.08	−0.168	−0.029	−0.228	−0.115	0.243	−0.233	−0.203	−0.251	−0.208	0.096
91	0.314	0.291	0.787	0.341	0.37	0.598	0.577	0.57	0.037	0.391	0.382	0.263
92	0.246	0.004	0.025	−0.161	0.01	−0.148	−0.116	−0.03	0.018	−0.127	−0.177	−0.138
93	0.105	0.126	0.107	−0.091	−0.105	−0.126	0.133	−0.013	0.011	0.005	−0.164	−0.03
94	0.133	0.135	0.264	−0.17	0.058	0.112	0	0.292	0.232	0.129	0.014	−0.006
95	0.385	0.433	0.543	0.042	0.167	0.293	0.675	0.646	0.55	0.62	0.42	0.361
96	0.053	0.105	0.319	−0.947	0.121	−0.179	−0.487	−0.214	0.226	−0.046	−0.299	−0.265
97	−0.2	0.109	−0.046	10.192	−0.305	−0.168	−0.06	0.259	−0.213	−0.088	−0.111	0

Table 5
Portfolios and Variances Based on ASX 8 Years Return Data

P	S_1	S_2	S_3	S_4-S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	Variance
P_{POS}	22.2	26.45	12.06		9.95	0.07		28.65		0.62	0.26
P_{FUZ}	59.11				12.10	6.19				22.60	0.03
P_{MAR}	43.10				33.77	12.58				10.55	0.04
P_{OWA}	36.20	14.66	6.68		14.25	3.78		15.87		8.56	0.15

Table 6
Realized Average Returns of the Portfolios Based on ASX Data (%)

Year(s)	P_{POS}	P_{FUZ}	P_{MAR}	P_{OWA}
1	13.432	9.556	9.351	11.672
2	30.833	25.035	28.315	28.752
3	19.584	11.304	8.524	15.463
4	12.644	4.887	3.919	9.035

average returns of P_{OWA} are better than those of P_{FUZ} and P_{MAR}, and slightly less than those of P_{POS}. The variance (risk) of P_{OWA} is greatly reduced (from 0.26 to 0.15) compared with that of P_{POS}.

Finally, different expected average return values (from 10% to 20%) were used to test the four portfolios based on the two sets of return data, the same conclusion was reached.

6.3. Flexibility and robustness testing

Our interest here does not reside in improving the performance of different models/techniques, but rather in how to integrate different models/techniques into one system under the unifying agent framework. Therefore the experiments conducted are focused on the integration, flexibility, and robustness.

First, the functions of different agents were tested to see whether they can work properly after being integrated together. To do this, the same input data sets were used as those in non-agent systems. The experimental results show that all agents in the prototype exhibit their behaviours correctly as the agent system can produce the same results as those of non-agent system.

The flexibility of the system was tested by launching the system first, and then adding new agents with similar capabilities to the system as well as deleting running ones from system. It was allowed these operations to be done on any host on the Internet. The testing process is as follows:

The serving agent (middle agent) is up and running first as all other agents in the system are required to register and connect to the serving agent. (To avoid single point failure, a replicated serving agent was used.) All other agents can then be started in any order. When an agent is started, it will send a KQML message to the serving agent to register its problem solving capabilities. The planning agent (together with the serving agent) coordinates the whole problem solving process. If no one agent with a specific problem solving ability (e.g. interest rate prediction or portfolio selection) is registered to the system, the system will display a message saying that no agent with a specific ability is available during the problem solving process. This is based on the meta knowledge built in the planning and serving agents. With all other agents and one interest rate prediction agent and one portfolio selection agent running, the system can provide results. We then start another interest rate prediction agent as well as the other two portfolio selection agents, the system can generate yet different results as usual.

With three portfolio selection agents running, we then delete (un-register) one or two of them, the system

can still work properly. The same holds for the interest rate prediction agents. For the decision aggregation agent, it can get information from the serving agent on how many portfolio selection agents were registered to the system. With this information, it can work out the final portfolio: If there is only one portfolio selection agent registered, it just returns that portfolio once received; If there are two or three portfolio selection agents registered, it will return the aggregated portfolio. Through the register and un-register process, the serving agent can keep updated information of other service provider agents.

By observing the performance of the system during the whole testing process, the system demonstrated very high flexibility and robustness. Through the testing process, it is also clear that we can add any more agents or delete agents with similar problem solving ability to the system at run time without affecting the functionality of the system.

In the implemented system, the serving agent can regularly send out "probing" KQML message to all registered agents. If no reply was received from a specific agent after the time-out, the serving agent will delete the registered information of that agent from its database. In this way, if one service provider agent disappears without un-registering itself from the serving agent (e.g. due to the crash of the computer running that agent or network failure), the serving agent can detect this and update its database accordingly.

To further test the robustness, a dedicated network was set up and the agent system prototype was run on the network environment. The network consists of hosts running different operating systems including Windows XP, Linux (RedHat), and Unix (Sun Solaris). The agents scattered on the hosts of the network and some of the service provider agents with similar problem solving ability were forced "out of order". In any of these situations, the system can still provide results, it is evident the system is robust.

7. Concluding remarks

Many real-world applications such as financial investment planning are very complicated and hybrid solutions are required. It is difficult to develop such hybrid intelligent systems since they have a large number of parts or components that have many interactions.

This paper argued that agent perspectives are suitable for building hybrid intelligent systems. An agent-based hybrid intelligent system for financial investment planning was developed. The system built from agent perspectives is flexible, robust, and interoperable. The success of the system indicates that agent technologies can significantly facilitate the building of loosely coupled hybrid intelligent systems. As different complementary techniques can be easily integrated into one system under the unifying agent framework, many complex problems such as financial investment planning can be solved in a shorter time frame and resulting in higher quality solutions.

While agent technology demonstrates the suitability of building hybrid intelligent systems, systematic methodologies need to be explored to analyze and design such agent-based hybrid intelligent systems.

Acknowledgement

This paper is partially supported by the Ministry of Education of China Key Project 104160 and Deakin University CRGS grant.

References

[1] J. Aczél and T.L. Saaty, Procedures for Aggregating Ratio Judgements, *Journal of Mathematical Psychology* **27** (1983).
[2] G. Bojadziev and M. Bojadziev, *Fuzzy Logic For Business, Finance, and Management*, World Scientific, Singapore, 1997.
[3] B. Bouchon-Meunier, (ed.), *Aggregation and Fusion of Imperfect Information*, Studies in Fuzziness and Soft Computing, Vol. 12, Physical-Verlag, 1998.
[4] V.K. Chaudhri, A. Farquhar, R. Fikes et al., OKBC: A Programmatic Foundation for Knowledge Base Interoperability, *Proc. 15th National Conference on Artificial Intelligence* (*AAAI'98*), AAAI Press/MIT Press, 1998, 600–607.
[5] K. Decker, K. Sycara and M. Williamson, Middle Agents for the Internet, *Proceedings of 15th International Joint Conference on Artificial Intelligence*, Nogoya, Japan, American Association for Artificial Intelligence, 1997, 578–583.
[6] M. Delgado, A.F. Gómez-Skarmeta et al., A Multi-Agent Architecture for Fuzzy Modeling, *International Journal of Intelligent Systems* **14** (1999), 305–329.
[7] D. Dubois and H. Prade, A Class of Fuzzy Measures Based on Triangular Norms, *International Journal of General Systems* **8**(1) (1982).
[8] M.R. Genesereth and S.P. Ketchpel, Software Agents, *Communications of ACM* **37**(7) (1994), 48–53.
[9] S. Goonatilake and S. Khebbal, (eds), *Intelligent Hybrid Systems*, Wiley, 1995.
[10] C. Iglesias, J. Gonzales and J. Velasco, MIX: A General Purpose Multiagent Architecture, in *Intelligent Agents II* (*ATAL'95*), *LNCS 1037*, Springer, 1996, 251–266.
[11] H.-A. Jacobsen, A Generic Architecture for Hybrid Intelligent Systems, in *Deep Fusion of Computational and Symbolic Processing*, Physica-Verlag, 2001, 145–173.
[12] N.R. Jennings, On Agent-Based Software Engineering, *Artificial Intelligence* **117**(1) (2000), 277–296.

[13] M. Knapik and J. Johnson, *Developing Intelligent Agents for Distributed Systems*, McGraw-Hill, New York, 1998.
[14] T. Finin, Y. Labrou and J. Mayfield, KQML as an Agent Communication Language, in: *Software Agents*, J.M. Bradshaw, ed., AAAI Press/ The MIT Press, Menlo Park, CA, 1997, pp. 291–316.
[15] R. Khosla and T. Dillon, *Engineering Intelligent Hybrid Multi-Agent Systems*, Kluwer Academic Publishers, Boston, 1997.
[16] M. Luck, P. Mcburney and C. Preist, A Manifesto for Agent Technology: Towards Next Generation Computing, *Autonomous Agents and Multi-Agent Systems* **9**(2004), 203–252.
[17] H. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, (2nd ed.), Blackwell, 1991.
[18] K. Sycara, K. Decker and D. Zeng, Intelligent Agents in Portfolio Management, in *Agent Technology: foundations, Applications, and Markets*, Springer, Berlin, 1998, 267–281.
[19] H. Tanaka, P. Guo and I. Turksen, Portfolio Selection Based on Fuzzy Probabilities and Possibility Distributions, *Fuzzy Sets and Systems* **111** (2000), 387–397.
[20] S.T. Welstead, *Neural Network and Fuzzy Logic Applications in C/C++*, Wiley, New York, 1994.
[21] M. Wooldrige, N. Jennings and D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, *Journal of Autonomous Agents and Multi-Agent Systems* **3**(3) (2000), 285–312.
[22] M. Wooldridge, *An Introduction to Mutliagent Systems*, John Wiley & Sons, Chichester, England, 2002.
[23] M. Wooldridge, Agent-Based Software Engineering, *IEE Proc. Software Engineering* **144**(1) (1997), 26–37.
[24] R.R. Yager, On Ordered Weighted Averaging Aggregation Operators in Multi-Criteria Decision Making, *IEEE Transactions on Systems, Man, Cybernetics* **18** (Jan/Feb 1988), 183–190.
[25] R.R. Yager and A. Rybalov, Uninorm Aggregation Operators, *Fuzzy Sets and Systems* **80** (1996), 111–120.
[26] R.R. Yager and J. Kacprzyk, eds, *The Ordered Weighted Averaging Operators: Theory and Applications*, Kluwer Academic, 1997.
[27] R.R. Yager, New Models of OWA Information Fusion, *Int J of Intelligent Systems* **13** (1998), 661–681.
[28] L.A. Zadeh, The Roles of Fuzzy Logic and Soft Computing in the Conception, Design and Deployment of Intelligent Systems, in *Software Agents and Soft Computing: Concepts and Applications, LNAI 1198*, Springer-Verlag, Berlin, 1997, 183–190.
[29] Z. Zhang and C. Zhang, Result Fusion in Multi-Agent Systems Based on OWA Operator, in *Proceedings of 23rd Australasian Computer Science Conference*, IEEE Computer Society Press, NJ, 2000, 234–240.
[30] Z. Zhang, *An Agent-Based Hybrid framework for Decision-Making on Complex Problems*, PhD Thesis, Deakin University, Australia, 2001.
[31] Z. Zhang and C. Zhang, An Agent-Based Hybrid Intelligent System for Financial Investment Planning, in *Proceedings of PRICAI 2002, LNAI-2417*, Springer, August 2002, 355–364.
[32] Z. Zhang and C. Zhang, An Improvement to Matchmaking Algorithms for Middle Agents. in *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, ACM Press, Italy, July 2002, 1340–1347.