AN AGENT-ORIENTED METHODOLOGY FOR HYBRID INTELLIGENT SYSTEM CONSTRUCTION

By

Chunsheng Li

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

University of Technology, Sydney

March 2005

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

_____

To My Wife Aiyun and Our Lovely Son Kan

Abstract

In recent years, both researchers and practitioners have recognised the advantages of applying the agent-based paradigm to the development of hybrid intelligent systems (HIS). Yet, the number of deployed commercial agent-based hybrid intelligent applications is small. One of the reasons for this is the lack of practical methodologies for agent-based hybrid intelligent applications development.

The aim of this thesis is to overcome this limitation. We have devised an agent-oriented methodology, called MAHIS (standing for ***M**ethodology for constructing **A**gent-based **H**ybrid **I**ntelligent **S**ystems*). To avoid building MAHIS from scratch, we have followed the strategy of extending an existing well-known methodology in order to bridge the gap between the existing methodology and agent-based HIS construction. MAHIS has extended the capabilities of MAS-CommonKADS in agent-based HIS development. It is suitable for constructing agent-based HIS, as well as analysing and designing any *open systems* with hierarchical structure. O*pen system* in this thesis means that the system allows for dynamic addition or removal of agents at run-time.

MAHIS consists of eight models: *Hybrid Strategy Identification Model*, *Organisation Model*, *Task Model*, *Agent Model*, *Expertise Model*, *Coordination Model*, *Reorganisation Model*, and *Design Model*. These models are grouped into three levels: *conceptualisation*, *analysis*, and *design*. Both the *Hybrid Strategy Identification Model* and *Reorganisation Model* are newly developed models rather than from MAS-CommonKADS. At the same time, some existing models have

been improved accordingly. The *Reorganisation Model* is the key model to support HIS and any *open system*s with hierarchical structure. It consists of *category role*, *group roles*, *virtual organisation role*, and *dynamics rules*. This model describes the hierarchical, dynamic, reusable, and unpredictable characteristics of HIS with *virtual organisation*, *categor*y, and *group* perspectives. Some previously developed agents can be reused by means of involving them in a new *virtual organisation* dynamically. The output of the *Reorganisation Model* is the specification of the dynamic platform which comprises middle agents and makes all agents and agent groups hierarchical and dynamic.

A dynamic platform PAHIS (***P****latform for* ***A****gent-based* ***H****ybrid* ***I****ntelligent* ***S****ystems*) has been developed. PAHIS not only verified the capability of MAHIS in dynamic platform construction, but also can be used as an infrastructure of agent-based HIS. It supports the dynamic addition and removal of group-based agents at run-time. A self-organising ring-based architectural model has been developed to organise middle agents in PAHIS. Moreover, the ring-based architectural model has been evaluated from the viewpoints of complexity, efficiency, extendibility, and availability. The ring-based architectural model is competent in agent-based systems with middle agents.

The verification of MAHIS in HIS construction has been conducted by two successful case studies: "financial investment planning system" and "petroleum reservoir characterisation system". The former has verified MAHIS in constructing agent-based HIS with tight-coupling hybrid strategy. The latter has verified MAHIS in constructing agent-based HIS with loose-coupling hybrid strategy. PAHIS has been employed in these two systems.

The evaluation of MAHIS with the enriched evaluation framework which was originally proposed by Cernuzzi and Rossi has been completed by comparing it with Gaia and MAS-CommonKADS. The *reorganisation attributes* have been added into the evaluation framework. The evaluation results have indicated that MAHIS is preferable over Gaia and MAS-CommonKADS, especially in the construction of agent-based HIS.

Acknowledgements

I would like to extend my immense gratitude to my supervisor Professor Chengqi Zhang and co-supervisor Dr. Zili Zhang for their guidance and stimulating discussions during the course of this work. They often offer me some constructive, insightful comments and helpful suggestions. They always give me timely feedback for my research ideas, papers and thesis drafts. I am lucky and happy to have been able to work with them during my PhD study. This thesis could not have been completed without their friendship, encouragement, and intellectual support.

I am grateful to the Faculty of Information Technology at University of Technology, Sydney (UTS) for providing me with an excellent environment, scholarship for my studies, and travel funds for my participation in both domestic and overseas conferences, which have broadened my horizon enormously.

I would like to express my deep appreciation to those who helped me in one way or another during my PhD study: namely Professor Yuzhi Yan, Dr. Rongmei Liu, Dr. Shichao Zhang, Mr. Longbing Cao, Dr. Qingfeng Chen, Professor Fenxi Zhang, Dr. Xiaowei Yan, Dr. Yujin Zhang, Ms. Mei Wang, Ms. Dan Cheng, Ms. Qingfeng Song, Ms. Li Liu, Ms. Ann Goldwater, Mr. Jiaqi Wang, Mr. Li Lin, Mr. Zhenxing Qin, Mr. Yanchang Zhao, and Mr. Jiarui Ni. I would also like to thank the anonymous examiners for their valuable comments and suggestions.

I would like to give my special thanks to my wife Aiyun and my son Kan for their support and understanding throughout the years when I was away from home for my PhD study in Australia.

List of the Publications

The following is a list of my research papers published in referred international conference proceedings or journals during my PhD study at University of Technology, Sydney (UTS):

1. Chunsheng Li, Zili Zhang, and Chengqi Zhang, A platform for dynamic organisation of agents in agent-based systems, in *Proceedings of the 2004 IEEE/WIC International Conference on Intelligent Agent Technology*, Beijing, China, 2004, 454-457.
2. Chunsheng Li, Li Liu, and Qingfeng Song, A practical framework for agent-based hybrid intelligent systems, *Asian Journal of Information Technology*, Vol. 3 (2), 2004, 107-114.
3. Chunsheng Li, Dan Cheng, and Chengqi Zhang, A platform to integrate well-log information applications on heterogeneous environments, in *Proceedings of the 2nd International Conference on Information Technology and Applications*, Harbin, China, 2004, 265-270.
4. Chunsheng Li, Qingfeng Song, and Chengqi Zhang, MA-IDS architecture for distributed intrusion detection using mobile agents, in *Proceedings of the 2nd International Conference on Information Technology and Applications*, Harbin, China, 2004, 451-455.
5. Chunsheng Li, Chengqi Zhang, and Longbing Cao, Theoretical evaluation of ring-based architectural model for middle agents in agent-based system, in

proceedings of the 14th International Symposium on Methodologies for Intelligent Systems, Maebashi, Japan, 2003, 603-607.

6. Chunsheng Li, Chengqi Zhang, Mei Wang, and Qingfeng Song, An approach to digitizing and managing well-logging graphs with agent-based perspective, in *Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology*, Halifax, Canada, 2003, 11-17.
7. Chunsheng Li, Qingfeng Song, Mei Wang, and Chengqi Zhang, SCTR: an approach to digitizing well-logging graph, in *Proceedings of the 6th IASTED International Conference on Computers, Graphics and Imaging*, Honolulu, USA, 2003, 285-288.
8. Longbing Cao, Chao Luo, Chunsheng Li, Chengqi Zhang, and Ruwei Dai, Open giant intelligent information systems and its agent-oriented abstraction mechanism, in *Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering*, San Francisco, USA, 2003, 85-89.
9. Longbing Cao, Chunsheng Li, Chengqi Zhang, and Ruwei Dai, Open giant intelligent information systems and its agent-oriented analysis and design, in *Proceedings of the 2003 International Conference on Software Engineering Research and Practice*, Vol. 2, CSREA Press, 2003, 816-822.
10. Chunsheng Li, Chengqi Zhang, and Mei Wang, An agent-based framework for well-logging information management, in *Proceedings of the 2nd International Conference on Active Media Technology*, Chongqing, China, 2003, 114-119.
11. Qingfeng Chen, Chengqi Zhang, Shichao Zhang, and Chunsheng Li, Verifying the Purchase Request in SET Protocol, in *Proceedings of the Fifth Asia Pacific Web Conference*, Xi'an, China, 2003, 263-274.
12. Chunsheng Li, Chengqi Zhang, and Mei Wang, An agent-based curve-digitizing system for well-logging data management, in *Proceedings of the International Conference on Information Technology: Coding and Computing*, Las Vegas, Nevada, USA, 2003, 656-660.

13. Chunsheng Li, Chengqi Zhang, and Zili Zhang, A ring-based architectural model for middle agents in agent-based system, in *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning*, Hong Kong, China, 2003, 94-98.

14. Chunsheng Li, Chengqi Zhang, Qingfeng Chen, and Zili Zhang, A scalable and robust framework for agent-based heterogeneous database operation, in *Proceedings of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, Vienna, Austria, 2003, 260-271.

15. Chunsheng Li, Mei Wang, and Liping Yang, An agent-based system for well-logging data manipulation on Intranet, *Daqing Petroleum Institute Journal*, 26(1), 2002, 54-56.

16. Chengqi Zhang, Chunsheng Li, and Zili Zhang, An agent-based framework for petroleum information services from distributed heterogeneous data resources, in *Proceedings of the 9th Asia Pacific Software Engineering Conference*, Gold Coast, Australia, 2002, 593-602.

17. Chunsheng Li, Chengqi Zhang, and Zili Zhang, An agent-based middleware for uniform operation in a heterogeneous database environment, in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, Vol. 1, Singapore, 2002, 385-389.

18. Chunsheng Li, Chengqi Zhang, and Zili Zhang, An agent-based intelligent system for information gathering from World Wide Web environment, in *Proceedings of the 2002 International Conference on Machine Learning and Cybernetics*, Vol. 4, Beijing, China, 2002, 1852-1857.

Contents

List of Tables

List of Figures

Chapter 1

1 Introduction

Many complex problems, such as financial investment planning and petroleum reservoir characterisation, involve many different components or sub-tasks, each of which requires different types of processing. To solve such complex problems, a great diversity of intelligent techniques are required, including traditional hard computing techniques and soft computing techniques (Medsker 1995). These techniques are complementary rather than competitive, and thus must be used in combination and not exclusively. These resulting systems are called *hybrid intelligent systems* (HIS) (Goonatilake and Khebbal 1995, Medsker 1995). In other words, hybrid solutions are crucial for complex problem solving. However, the design and development of HIS are difficult because they involve a large number of parts or components that have many interactions. Existing software development techniques cannot manage those complex interactions efficiently as these interactions may occur at unpredictable times, for unpredictable reasons, and between unpredictable components (Zhang and Zhang 2004).

In recent years, both researchers and practitioners have recognised the advantages of applying the agent-based paradigm for the development of HIS (Khosla and Dillon 1997, Centeno-Gonzalez, Velasco et al. 1999, Li, Liu et al. 2004, Zhang and Zhang 2004). Agent techniques represent an exciting new means

of analysing, designing and building complex software systems. An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives (Wooldridge 1997, Jennings 2000). They have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasible be tackled (Jennings 2000). A multi-agent system (MAS) is a collection of autonomous agents that work together to solve the problems that are beyond the capabilities of individual agents. The multi-agent perspective has distinct advantages in *decomposition*, *abstraction*, and *organisation*, which are the fundamental tools of the trade for helping to manage complexity of HIS (Jennings 2001, Zhang and Zhang 2004). Thus agent perspective is suitable for modelling, designing, and constructing HIS.

Yet, the number of deployed commercial agent-based hybrid intelligent applications is small. One of the reasons for this is the lack of practical methodologies for agent-based hybrid intelligent applications development. Even in the agent research field, although more than two dozens agent-oriented methodologies have been developed during the last decade (Sturm and Shehory 2003), only a few complete and well-grounded methodologies have been proposed to the analysis and design of MAS so far (Zhang 2001, Cuesta, Gomez et al. 2004). Moreover, these methodologies are deficient in HIS construction because they did not take into account the characteristics of HIS. HIS has their own distinct characteristics, such as, the hierarchical structure of inter-related subsystems, the arbitrary primitive components of subsystems, the dynamic components of system, and the unpredictable interactions among these components, rather than general agent-based systems. Although some researchers have considered this issue and attempted to propose agent-oriented methodologies for constructing HIS (Khosla and Dillon 1997, Srikanth 1999, Zhang and Zhang 2004), there is no one methodology that can fully meet the requirements of the analysis and design of agent-based HIS (Zhang and Zhang 2004).

Analysing the problems within the domain of agent-oriented methodologies for constructing HIS, we have at first clarified the gap between the existing well-known

agent-oriented methodologies and agent-based HIS construction by means of evaluating these methodologies. For comparing and evaluating the existing well-known agent-oriented methodologies, we have employed and enriched the evaluation framework proposed by Cernuzzi and Rossi (Cernuzzi and Rossi 2002). This framework evaluates agent-oriented methodologies with a qualitative analysis followed by a quantitative rating. It is very convenient to extend this evaluation framework with the characteristics of HIS. We have enriched the framework with *reorganisation attributes* which are related to the distinct characteristics of HIS.

Six well-known agent-oriented methodologies have been analysed and ranked based on the enriched evaluation framework. The evaluation results have shown that the existing agent-oriented methodologies are deficient in the *reorganisation attributes*. Only MAS-CommonKADS (Iglesias, Garijo et al. 1996, Iglesias, Garijo et al. 1997) has the capability of supporting the *organisational structure* and *coordination mechanism* which are part of the *reorganisation attributes*. At the same time, MAS-CommonKADS is better than other methodologies, after ranking these methodologies according to the attributes related to each hybrid strategy. To avoid building the agent-oriented methodology for constructing HIS from scratch, MAS-CommonKADS is selected to be tailored (extended and cut) in order to bridge the gap between MAS-CommonKADS and agent-based HIS construction.

Following the tailored methodology, MAHIS (***M****ethodology for constructing* ***A****gent-based* ***H****ybrid* ***I****ntelligent* ***S****ystems*) has been proposed. MAHIS has extended the capabilities of MAS-CommonKADS in agent-based HIS development. It is suitable for constructing agent-based HIS, as well as analysing and designing any *open systems* with hierarchical structure. O*pen system* in this thesis means that the system allows for dynamic addition or removal of agents while multi-agent system is running (Cuesta, Gomez et al. 2004).

Some related issues including MAHIS modelling, dynamic platform development, and verification and evaluation of MAHIS have been discussed. MAHIS has been verified by a dynamic platform PAHIS (***P****latform for* ***A****gent-based* ***H****ybrid* ***I****ntelligent* ***S****ystems*) and two case studies: "financial investment planning system" and "petroleum reservoir characterisation system". The evaluation results

have shown that MAHIS is preferable over other methodologies in agent-based HIS construction.

This chapter is the overview of the thesis. It briefly covers the characteristics of the hybrid strategies, the gap between the existing well-known agent-oriented methodologies and agent-based HIS construction, the outline of the proposed MAHIS methodology, the verification and evaluation of MAHIS, and the contributions of this thesis. The specific organisation of this chapter is as follows. Section 1.1 is about the hybrid strategies and their characteristics. Section 1.2 clarifies the gap between the existing well-known agent-oriented methodologies and agent-based HIS construction. Section 1.3 introduces the proposed methodology MAHIS. Section 1.4 discusses the related issues during MAHIS construction. Section 1.5 presents what has been achieved in this thesis. Finally, Section 1.6 describes the organisation of the following chapters.

1.1 Characteristics of Hybrid Strategies

The intelligent systems using expert systems (*ES*), neural networks (*NN*), fuzzy logic systems (*FL*), genetic algorithms (*GA*), case-based reasoning (*CBR*), and so on, respectively, have accomplished substantial gains, but there are problems associated with them too. The modern trend is to integrate those individual intelligent techniques by using hybrid strategies for offsetting the demerits of one technique by the merits of another one. The resulting systems are called HIS (Medsker 1995).

Many hybrid strategies and implementation techniques have been proposed for solving special kinds of complex problems. However, the research on the theoretical modelling of the practical hybrid strategies is not enough to cover the gamut of work on the HIS construction. In the HIS community, researchers mainly focus on *hybrid techniques*, *hybrid strategies*, and *hybrid modelling*. The *hybrid modelling* focuses on the models of hybrid techniques and hybrid strategies for abstracting and formalising the techniques and strategies in hierarchy. In the *hybrid techniques*, researchers attempt to find some possible efficient methods according to

the requirements of specific problems. Each hybrid technique combines two or more intelligent techniques by using relations ⫫ (*COMBINATION*), ⫦ (*TRANSFORMATION*), and Θ (*FUSION*) (Khosla and Dillon 1997). Relation ⫫ denotes that knowledge representation and processing of two intelligent techniques are fully interleaved. Relation ⫦ denotes that knowledge representation of one intelligent technique is transformed into another one's. Relation Θ denotes that two intelligent techniques use the same knowledge representation, but the centre around intelligent technique is the former.

The *hybrid strategy* (also called *hybrid integration strategy*) model focuses on different levels of intelligent activities and systems, and the individual technologies and their hybrids. Issues range from the fundamental questions about the nature of cognition and theories of computation to problems of exactly how best to implement hybrid systems. Five different hybrid strategy models have been identified: *stand-alone model*, *transformational model*, *loose-coupling model*, *tight-coupling model*, and *fully-integration model* (Medsker 1995). The *stand-alone models* consist of independent software components. These components do not interact in any way. The stand-alone approach uses independent systems to study different or the same aspects of an application problem with no prior commitment for the implementation technique of the final operational system. Exploratory work with each system can lead to a better understand of an application and verify knowledge to be incorporated into an intelligent system. The structure of a stand-alone system consists of three levels: *subsystems*, *results identifier*, and *results mediator*. A task is synchronously completed by all subsystems using different stand-alone intelligent techniques. The outputs of the subsystems are identified by the results identifier. The results mediator makes decision by analysing these identified results.

The *transformational models* are similar to the stand-alone models in that the end result of development is an independent model that does not interact with the other. What distinguishes the two types of models is that transformational systems begin as one type of system, and end up as the other. The *TRANSFORMATION* (⫦) and *FUSION* (Θ) are related to *transformational hybrid strategy models*. The

structure of a transformational system consists of two levels: subsystems and results mediator. Compared with the stand-alone systems, the results identifier is not useful in the transformational system because the subsystems themselves have transformed their results into the outputs in a uniform fashion.

The *loose-coupling models* are the first true form of integrated HIS. The application is decomposed into separate intelligent components that communicate *via* data files. The variations of the *loose-coupling models* are pre-processors, post-processors, co-processors, and user interfaces. The *TRANSFORMATION* (╠) is related to the *loose-coupling hybrid strategy models*. The structure of a loose-coupling system consists of three levels: components, manager, and application. The manager level consists of task delegation and data management. The application level includes data files. All components cooperate to complete the task. Each component which consists of an intelligent technique or a hybrid technique interacts with other ones by means of data files. The data files are manipulated by the data management.

The categories of loose- and tight-coupling have significant overlap. Both utilise independent intelligent technique *x* and intelligent technique *y*. However, tight-coupling systems pass information *via* memory resident data structures rather than external data files. This improves the interactive capabilities of tight-coupling models in addition to enhancing their performance. The *TRANSFORMATION* (╠) is related to *tight-coupling hybrid strategy models*. The structure of a tight-coupling system consists of three levels: components, manager, and application. The manager level consists of task delegation and message management. The application level includes messages. Compared with the loose-coupling systems, the difference is that each component in tight-coupling system interacts with other ones by means of messages rather than data files. The messages are organised by the message management.

The *fully-integration* systems share data structures and knowledge representations. Communication between the different components is accomplished *via* the dual nature of the structures. The *COMBINATION* (╬) is related to the *fully-integration hybrid strategy models*. The structure of a fully-integration system

consists of three levels: components, manager, and application. The manger level consists of knowledge base manipulator and task delegation. The application level includes knowledge base. Compared with the loose-coupling and tight-coupling systems, the difference is that all components in a fully-integration system share a uniform knowledge base instead of each component with its own knowledge base.

From the aforementioned characteristics of the hybrid strategies adopted in HIS, the characteristics of HIS are summarised as following (Zhang and Zhang 2004):

- HIS consists of a number of inter-related subsystems or components organised in a hierarchical fashion.
- The choice of the primitive components in each hierarchical level is arbitrary and is defined according to the needs of the observers.
- The primitive components in each hierarchical level may be dynamic at unpredictable time.
- The interactions between primitive components may occur at unpredictable times and for unpredictable reasons.

1.2 Methodologies for Agent-Based HIS

Agent perspective is suitable for modelling, designing, and constructing HIS (Jennings 2001, Zhang and Zhang 2004). From the exempla of developed agent-based HIS (Iglesias, Centeno-Gonzalez et al. 1995, Scherer and Schlageter 1995, Khosla and Dillon 1997, Iglesias, Centeno-Gonzalez et al. 1998, Jacobsen 1998, Lertpalangsunti and Chan 1998, Centeno-Gonzalez, Velasco et al. 1999, Delgado and Gomez-Skarmeta 1999, Zhang and Zhang 2000a, Zhang and Zhang 2000b, Zhang 2001, Luo, Zhang et al. 2002, Li, Liu et al. 2004, Zhang and Zhang 2004), it is known that agent-based HIS were usually developed from scratch rather than by following an agent-oriented methodology because no one methodology can fully meet the requirements of the analysis and design of agent-based HIS (Zhang and Zhang 2004). So, it is one of the urgent tasks in agent-based HIS field to propose a

competent methodology. The objective of this thesis is to build a methodology for agent-based HIS construction.

To avoid building the *new methodology* from scratch, we have followed the following research strategy. Firstly, clarify the gap between the existing well-known agent-oriented methodologies and agent-based HIS construction by means of evaluating these methodologies based on the characteristics of HIS. At the same time, select a better methodology from the evaluated well-known methodologies as the candidate to be tailored. The word "*tailor*" in this thesis means "*extend*" and "*cut*". Secondly, propose the *new methodology* by extending the *selected methodology* in order to bridge the gap between the *selected methodology* and agent-based HIS construction. At the same time, some redundant contents of the *selected methodology* are cut out because they are not suitable for constructing HIS, or they conflict with the extended parts. Thirdly, verify the capabilities of the *new methodology* by case studies. Finally, evaluate the *new methodology* with the evaluation framework which has taken the characteristics of HIS into account.

The following subsections discuss the first step of this research strategy.

1.2.1 Evaluation Criteria with HIS Attributes

Some evaluation frameworks for agent-oriented methodologies have been proposed (Bourne, Shoop et al. 2001, O'Malley and Deloach 2001, Shehory and Sturm 2001, Cernuzzi and Rossi 2002, Dam and Winikoff 2003, Mali 2003, Cuesta, Gomez et al. 2004, Sudeikat, Braubach et al. 2004). They cannot directly be employed to evaluate methodologies for agent-based HIS construction because agent-based HIS possesses their distinct characteristics as described in Section 1.1. However, these evaluation frameworks can be used to evaluate the existing agent-oriented methodologies if they are enriched with the attributes of agent-based HIS. We have employed and enriched the evaluation framework proposed by Cernuzzi and Rossi (Cernuzzi and Rossi 2002) for comparing and evaluating the existing agent-based methodologies. The framework is selected based on the following reasons. Firstly,

this framework is known as a qualitative analysis followed by a quantitative rating. Secondly, the extension of the evaluation criteria is convenient. The significance of the framework is the construction of an *attributes tree*, where each node of the tree represents a software engineering criterion or a characteristic of agent-based system. The *attributes tree* organises the found criteria in weight branches. To extend the *attributes tree* just adds some defined roots or braches. Finally, this framework is cited by some valuable evaluation frameworks (Dam and Winikoff 2003, Sturm and Shehory 2003, Sudeikat, Braubach et al. 2004).

The evaluation framework proposed by Cernuzzi and Rossi is enriched with *reorganisation attributes* according to the characteristics of HIS. In the extended evaluation framework, the attributes are grouped into four different categories: those concerning the own characteristics of agents (*internal attributes*), those referred to the interaction process (*interaction attributes*), those standing for the reorganisation of agents (*reorganisation attributes*), and those more directly inherent to the design and development process (*other process requirements*). The *internal attributes* include *autonomy*, *reactivity*, *pro-activeness*, and *mental notions*. The *interaction attributes* include *social ability*, *interaction with the environment*, *multiple control*, *multiple interests*, and *subsystems interaction*. The *other process requirements* include *modularity*, *abstraction*, *system view*, and *communication support* (Cernuzzi and Rossi 2002).

The reorganisation attributes include *hierarchical structure*, *agent as a level member*, *dynamic agents in each level*, *application-based reorganisation*, and *shared items management*. The hierarchical structure includes three attributes: *agent categories*, *organisation structure*, and *coordination mechanism*. An agent category consists of a set of isomorphic agents. The organisational structure organises all agents together. The coordination mechanism decides how agents or agent groups interact under the organisational structure. The "*agent as a level member*" indicates the type of the primitive members in each hierarchical level. The "d*ynamic agents in each level*" indicates the mechanism that the primitive members of each hierarchical level can be dynamically added or removed at run-time. The "*application-based reorganisation*" indicates that agents located in different

categories can be dynamically organised as an organisation. The "s*hared items management*" includes three attributes: data, legacy systems, and agents. The data may include databases, data files, websites, etc. The legacy system indicates that application is not based on agent technique.

During evaluation of methodologies, each attribute is assigned with a score and the score of attributes on the node is calculated based on those of their children. The score assigned to each attribute is in the range of 0 to 10.

1.2.2 Evaluating Methodologies with Criteria

Six well-known agent-oriented methodologies, Gaia (G) (Wooldridge, Jennings et al. 2000, Zambonelli, Jennings et al. 2003), MAS-CommonKADS (Mc) (Iglesias, Garijo et al. 1996), MaSE (Ms) (Deloach, Wood et al. 2001), ODAC (O) (Gervais 2003), Prometheus (P) (Padgham and Winikoff 2002, Padgham and Winikoff 2003), and Tropos (T) (Bresciani and Giorgini 2002, Mouratidis, Giorgini et al. 2002, Giunchiglia, Mylopoulos et al. 2003), have been compared with the enriched evaluation criteria. These methodologies were selected since they (a) were described in more detail (e.g. journal paper rather than a conference paper); and (b) were perceived as being significant by the agent community. Another important factor was whether the methodology had been developed over an extended time period based on feedback from users other than the developers of the methodology (Dam and Winikoff 2003). The evaluation results are presented in Table 1.1 (see subsection 3.3.1 for details). The number in Table 1.1 is the average of all attributes' values in an attribute category against a methodology.

From the evaluation results, we know that all methodologies are deficient in *reorganisation attributes* which denote the capability of a methodology to construct HIS. Only MAS-CommonKADS meets the *organisation structure* and *coordination mechanism* attributes which are the children of *hierarchical structure* node, so the value of the *reorganisation attributes* is assigned 1.33. However, all methodologies are strong in the *internal attributes* and *other process requirements*. MAS-

CommonKADS and Prometheus are better in the *internal attributes*. MAS-CommonKADS is better than others in the *interaction attributes*. MAS-CommonKADS and MaSE are better than other methodologies in the *other process requirements*. From the comprehensive attributes viewpoint, MAS-CommonKADS is better than others.

Table 1.1 Evaluation results of methodologies

Attribute Categories (Value)	G	Mc	Ms	O	P	T
Internal Attributes (10)	7.83	***8.33***	***8.1***	7.6	8.6	7.4
Interaction Attributes (10)	4.7	***7***	4.9	3.9	4.3	3.3
Reorganisation Attributes (10)	0	1.33	0	0	0	0
Other Process Requirements (10)	7.5	***9.5***	***9.5***	8.1	8.3	7.9
Accumulative Total (40)	**20**	**26.2**	**22.5**	**19.6**	**21.2**	**18.6**

1.2.3 Methodology Selection for Tailoring

The relationships between the evaluation attributes and the hybrid strategies have been analysed according to the characteristics of each hybrid strategy. The five attributes, namely, *hierarchical structure*, *agent as a level member*, *dynamic agents in each level*, *application-based reorganisation*, and *shared items management*, evaluate the capabilities of a methodology in constructing agent-based HIS with dynamic organisation and hierarchical structure in accordance with the characteristics of HIS. The *stand-alone* hybrid strategy associates the following attributes: *autonomy, reactivity, mental notions, interaction with the environment, hierarchical structure, agent as a level member, dynamic agents in each level, and other process requirements* according to the characteristics of the stand-alone systems. The transformational hybrid strategy associates the following attributes: *autonomy, reactivity, mental notions, interaction with the environment, multiple control, multiple interests, hierarchical structure, agent as a level member, dynamic agents in each level, and other process requirements*. The loose-coupling hybrid strategy associates all attributes. The tight-coupling hybrid strategy associates the

following attributes: *internal attributes, social ability, interaction with the environment, multiple control, reorganisation attributes, and other process requirements*. The fully integration hybrid strategy associates the following attributes: *autonomy, reactivity, mental notions, social ability (commitments), interfaces with other entities, interaction with the environment, reorganisation attributes, and other process requirements*.

The methodologies for constructing HIS with different hybrid strategy model are ranked in Table 1.2 (see subsection 3.3.2 for details). The number in Table 1.2 is the average of the attributes' values associated a hybrid strategy model. From Table 1.2 we know that MAS-CommonKADS is better than other methodologies for constructing all kinds of HIS. So MAS-CommonKADS can be selected as the candidate to be tailored in order to propose the new methodology (MAHIS) for agent-based HIS construction.

Table 1.2 Ranking results of the methodologies

Hybrid Strategy Models	G	Mc	Ms	O	P	T
Stand –alone strategy	5.10	**6.19**	5.02	4.64	5.41	4.70
Transformational strategy	4.91	**5.96**	4.45	3.84	4.46	4.12
Loose-coupling strategy	5.00	**6.54**	5.63	4.90	5.30	4.65
Tight-coupling strategy	5.61	**6.53**	4.98	4.17	4.70	4.48
Fully Integration strategy	6.98	**8.02**	6.28	5.89	6.66	5.33

1.3 Outline of MAHIS Methodology

MAHIS attempts to direct users to develop their HIS from the descriptions of the problems to the output of the specifications which can be implemented directly. Two distinctive goals have been achieved after proposing and developing MAHIS based on MAS-CommonKADS. The first one is to extend MAS-CommonKADS for bridging the gap between MAS-CommonKADS and the HIS construction. The second is to cut out the redundant contents of MAS-CommonKADS that are not suitable for constructing HIS, or in conflict with the extended parts. We have

tailored MAS-CommonKADS in the MAHIS construction. Although MAHIS employed some techniques of MAS-CommonKADS, it is an independent and new methodology for constructing agent-based HIS.

MAHIS consists of eight models: *Hybrid Strategy Identification Model*, *Organisation Model*, *Task Model*, *Agent Model*, *Expertise Model*, *Coordination Model*, *Reorganisation Model*, and *Design Model*. Both the *Hybrid Strategy Identification Model* and *Reorganisation Model* are newly developed models. At the same time, some other existing models in MAS-CommonKADS have been improved accordingly. The relationships between those models are presented in Figure 1.1.

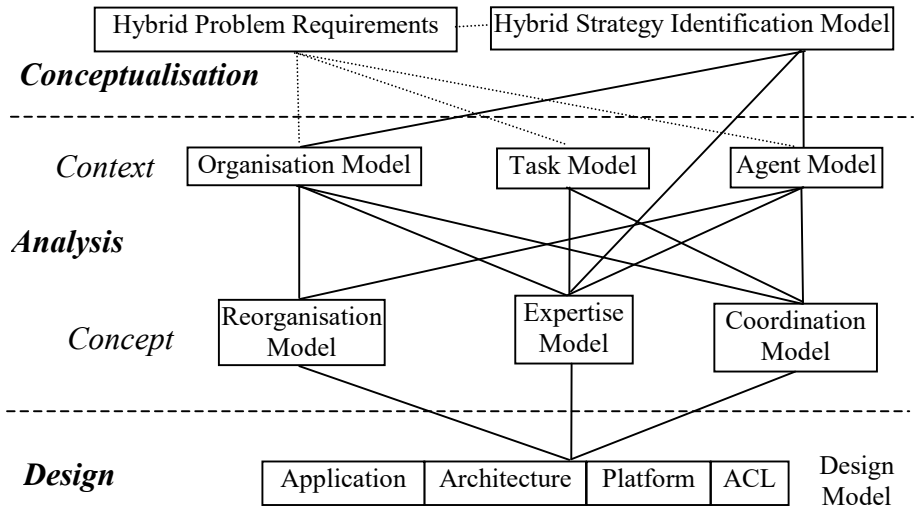

Figure 1.1 Framework of MAHIS methodology

The models of MAHIS are grouped into three levels: conceptualisation level, analysis level, and design level in accordance with the three process stages of MAHIS. The *conceptualisation level* includes the *Hybrid Strategy Identification Model* and the description of hybrid problem requirements. During this phase, an elicitation task to obtain a preliminary description of the hybrid problem is carried out. Based on the problem description, the hybrid strategy model adopted by the HIS is identified. The purpose to identify the hybrid strategy is to help other models to decide the architectural model of the HIS because the hybrid strategy adopted by a hybrid intelligent system decides the organisational structure and coordination

mechanism of the system. The analysis level includes the *Organisation Model*, *Task Model*, *Agent Model*, *Reorganisation Model*, *Expertise Model*, and *Coordination Model*. Those models can be divided into two sublevels: context and concept. The context sublevel includes the *Organisation Model*, *Task Model*, and *Agent Model*, which attempt to clarify the tasks, agents, organisational context, and environment. The concept sublevel includes the *Reorganisation Model*, *Expertise Model*, and *Coordination Model*, which issue the conceptual descriptions of the knowledge applied in a task, the interactions between agents, and the hierarchical structure and the primitive members in each level. The design level only includes the design model which consists of four steps: architecture design, agent communication language (ACL) design, platform design, and application design.

The hybrid problem to be solved is represented in the *Hybrid Problem Requirements*. The information in the *Hybrid Problem Requirements* can be used to develop the *Hybrid Strategy Identification Model*, *Organisation Model*, *Task Model*, and *Agent Model*. The *Organisation Model* supports the analysis of the major features of an organisation in order to describe the organisation into which HIS and the social organisation of the agent society are to be introduced. The *Task Model* analyses the global task layout, its inputs and outputs, preconditions and performance criteria, as well as needed resources and competences. The *Agent Model* describes the agent characteristics: groups and hierarchy. The purpose of the *Expertise Model* is to explicate in detail the types and structures of the knowledge used in performing a task. The *Coordination Model* describes the conversations between agents: their interactions, protocols and required capabilities. The *Reorganisation Model* is the key model to support HIS and *open system*s with hierarchical structure. It consists of *category role*, *group roles*, *virtual organisation role*, and *dynamics rules*. This model describes the hierarchical, dynamic, reusable, and unpredictable characteristics of HIS with *virtual organisation (VO)*, *categor*y, and *group* perspectives. The *VO* in this thesis is regarded as a running subsystem or application. The members of a *VO* include all agents for completing the task of the subsystem or application. An agent may belong to more than one *VO* at the same time. Some previously developed agents can be reused by means of involving them

in a new *VO* dynamically. The output of the *Reorganisation Model* is the specification of the dynamic platform which comprises middle agents (Decker, Sycara et al. 1997, Hanachi and Sibertin-Blanc 2004) and makes all agents and agent groups hierarchical and dynamic. The *Design Model* gives the technical system specification in terms of application, architecture, platform, and agent communication language to concretise the outputs of the *reorganisation*, *coordination*, and *expertise models*. The output of the design model can be implemented based on the different developing environments.

MAHIS has three distinct characteristics which are not covered by other agent-oriented methodologies. Firstly, MAHIS is suitable for constructing agent-based HIS as well as any open systems with hierarchical structure. Secondly, MAHIS supports the construction of agent-based systems with the ability of reorganisation of agents. Finally, dynamic platform development is taken into account from the methodology point of view. The platform can dynamically organise all agents in a system.

1.4 Key Research Issues of MAHIS

From the description in the above section, we have known some concepts of MAHIS methodology, for example, models, process stages, the capabilities of the models, architectural model, and so on. However, some issues like a full lifecycle process, a full set of techniques, a modelling language, etc. are still left to be tackled. These issues can be solved in *MAHIS modelling*. Furthermore, the competency of MAHIS needs to be verified by case studies and evaluated by means of evaluation criteria.

1.4.1 MAHIS Modelling

A methodology should include three aspects of components: lifecycle process, technique set and modelling language (Sturm and Shehory 2003). A fully agent-

oriented lifecycle modelling framework can provide a set of activities to construct and manage HIS in several procedures. The technique set includes techniques or methods which complete the tasks of a proposed system. They include the techniques that have been tried and tested over the last decade; but also may include new techniques that are more experimental (Graham, Henderson-Sellers et al. 1997). Some indication on the level of maturity of the individual technique is thus given as part of its full specification. The modelling language consists of meta-model and notation adopted by a methodology. A methodology needs to include a means for representing the generated artefacts; it needs to contain a notational element. Figure 1.2 shows the components of MAHIS and the relationships among them.

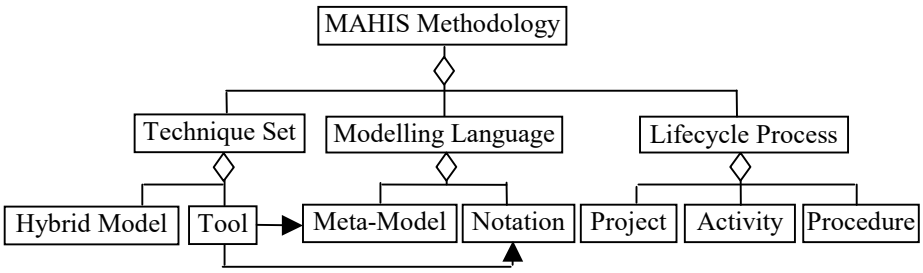

Figure 1.2 The components of MAHIS

A hybrid system development life cycle (HSDLC) for developing agent-based HIS has been proposed (see Section 4.1). The HSDLC consists of a set of activities, such as, *hybrid technique and hybrid strategy analysis*, *agent categorisation with hierarchical structure*, *platform development*, and *application development*. Those activities are grouped into five process stages (procedures): conceptualisation, analysis, design, implementation, and maintenance. The *platform development* and *application development* are separated for supporting dynamic addition and removal of agents and applications (projects), which make the HSDLC support dynamic project management. While the models of MAHIS were proposed, some meta-model with well-known notations, such as AUML (Bauer, Muller et al. 2001), use case of OOSE (Iglesias, Garijo et al. 1997), MSC and HMSC (Rudolph, Graubman et al. 1996), SDL (Iglesias, Garijo et al. 1997), KQML (Finin, Labrou et al. 1997, Wooldridge 2001), DFD (Data Flow Diagraph) (Mrhailaf and Sahraoui 1993), and

CDL (Schreiber, Akkermans et al. 1999), have been adopted for representing the generated HIS. The contents and the formal presentation of the outputs of each model have been defined and developed (see Chapter 4). About the techniques, MAHIS supports the hybrid strategies described in Section 1.1 and the hybrid techniques which have been developed or will be developed. At the same time, some tools based on the meta-models can be employed to represent the notations.

1.4.2 Verification of MAHIS in Dynamic Platform Construction

Platform is one of the crucial parts in agent-based HIS because it makes agents and applications dynamic. MAHIS has the capability to guide the developers to construct the dynamic platform. The *reorganisation model* and part of the *design model* in MAHIS are in charge of the construction of dynamic platform for agent-based HIS. The *category role*, *group roles*, *virtual organisation role*, and *dynamics rules* in the *reorganisation model* complete the analysis of the platform. The *category role* indicates the rules to categorise agents in the systems. At the same time, the categories which act as the category role are described. The group roles depict the primitive members of each category and the rules to organise a group. The *virtual organisation role* depicts the rules to form a VO dynamically. The dynamics rules describe the mechanisms which add agents into or remove them from the platform or a VO dynamically. The organisational structure and coordination mechanism of the platform design step in the *design model* complete the design phase of the dynamic platform. The development of platform and applications has been separated in the SHDLC. This is wise because platform as the infrastructure of a system must be developed first.

A dynamic platform PAHIS has been developed with middle agents by following MAHIS. PAHIS not only verified the capability of MAHIS in the platform construction, but also can be used as an infrastructure of agent-based HIS. Multiple middle agents (Hanachi and Sibertin-Blanc 2004) in PAHIS are organised with the

ring architectural model which is evaluated to be powerful and efficient by performance predictability, adaptability, and availability (Li, Zhang et al. 2003a). They dynamically manage the registration and cancellation of service provider agents and application agent groups.

1.4.3 Verification of MAHIS in HIS Construction

The best way to verify the competency of a methodology is to develop different types of applications by following the methodology (Fisher and Wooldridge 1997). From this point of view, we have successfully developed two applications: "financial investment planning system" and "petroleum reservoir characterisation system" for verifying MAHIS in HIS construction. The former system has verified MAHIS in constructing agent-based HIS with tight-coupling hybrid strategy. The latter system has verified MAHIS in constructing agent-based HIS with loose-coupling hybrid strategy. PAHIS with middle agents has been employed in those two systems.

The financial investment planning system gives advice to the investors. Diverse intelligent techniques have been integrated in this system. The intelligent techniques include: financial risk tolerance mode based on fuzzy logic, asset allocation model based on fuzzy logic, portfolio selection models (Markowitz's model, fuzzy probability portfolio selection model, and possibility portfolio selection mode), interest prediction models (feed forward network model, and combination of fuzzy logic and genetic algorithms model), ordered weighted averaging (OWA) operators for result aggregation, and expert system with explanation mechanisms. The exchanging information between the agents in this system is messages which can be directly carried by KQML (Finin, Labrou et al. 1997).

The petroleum reservoir characterisation system has integrated four intelligent technique agents (complicated lithology identification with parthenogenetic algorithm, porosity prediction with neural network, permeability estimation with fuzzy neural network, and well logs curve-digitizing with expert system). The

framework of the petroleum reservoir characterisation system consists of three categories of agents: application agent category, middle agent category, and service provider category. There are four agent groups in the *service provider category*: intelligent technique agent, curve digitisation agent, decision aggregation agent, and middleware agent (Li, Zhang et al. 2002a). The exchanging information between the agents in this system is data which may be included in data files or databases. The middleware agents are in charge of the management of these data resources. For supporting data exchange, a database-based agent communication language (DBACL) (Li, Zhang et al. 2002a) has been proposed based on KQML. Under the support of PAHIS, the prototype of the petroleum reservoir characterisation system is implemented using C language and Socket technique.

1.4.4 Evaluation of MAHIS

The evaluation of MAHIS with the enriched evaluation framework described in Section 1.2.1 has been conducted by comparing MAHIS with Gaia and MAS-CommonKADS. MAHIS has been evaluated by following the steps: *application of the paradigm Goal-Question-Metric (GQM)*, *specification of an attributes tree*, *definition of the empiric relationships, and evaluation of qualitative and quantitative attributes*. From the evaluation results, it is quite evident that MAHIS in all perspectives presents better competency than Gaia and MAS-CommonKADS in agent-based HIS construction. Moreover, in the *reorganisation attributes* perspective the difference is very pronounced because MAHIS supports the construction of *open systems* with hierarchical structure.

1.5 Principal Contributions of the Thesis

In this thesis an agent-oriented methodology called MAHIS is proposed by tailoring MAS-CommonKADS for guiding developers to construct agent-based HIS. MAHIS is suitable for constructing agent-based HIS as well as any *open systems*

with hierarchical structure. In addition, MAHIS supports the construction of agent-based systems with the capability of agent reorganisation by means of dynamic platform construction which is taken into account in all process stages of MAHIS. For constructing MAHIS, some relevant works have been conducted. Firstly, the gap between the existing well-known agent-oriented methodologies and agent-based HIS construction has been clarified by evaluating these methodologies with the *attributes tree*. We have enriched the *attributes tree* according to the characteristics of HIS, and ranked these methodologies based on the *attributes tree*. Secondly, a platform called PAHIS, which supports the dynamic addition and removal of group-based agents at run-time, has been developed as a case study to verify MAHIS in platform development. A self-organising ring-based architectural model has been proposed to organise the middle agents in PAHIS. Thirdly, two case studies, "financial investment planning system" and "petroleum reservoir characterisation system", have been developed to verify MAHIS in application development. Finally, MAHIS has been evaluated by comparing it with Gaia and MAS-CommonKADS. The evaluation results have shown that MAHIS is preferable over other methodologies.

The main contributions in this thesis are summarised as following:

- The gap between the current existing well-known agent-oriented methodologies and agent-based HIS construction has been clarified by evaluating these methodologies with the framework proposed by Cernuzzi and Rossi (Cernuzzi and Rossi 2002). In this research, the following additional achievements have been obtained. Firstly, the characteristics of HIS have been extracted after modelling the hybrid techniques and hybrid strategies. Secondly, the attributes of the evaluation framework have been enriched in accordance with the characteristics of HIS. Finally, six existing well-known agent-oriented methodologies, namely, Gaia, MAS-CommonKADS, MaSE, ODAC, Prometheus, and Tropos, have been ranked based on the attributes associated each hybrid strategy. MAS-CommonKADS has been selected as the candidate to be *tailored* for constructing MAHIS.

- MAHIS methodology has been proposed by extending MAS-CommonKADS. Both *Hybrid Strategy Identification Model* and *Reorganisation Model* are newly developed models rather than from MAS-CommonKADS. At the same time, some other existing models have been improved accordingly. The *Reorganisation Model* is the key model to support HIS and any *open system*s with hierarchical structure. This model describes the hierarchical, dynamic, reusable, and unpredictable characteristics of HIS with *virtual organisation*, *category*, and *group* perspectives. Some previously developed agents can be reused by means of involving them in a new *virtual organisation* dynamically. Moreover, a hybrid system development life cycle (HSDLC) followed by MAHIS has been presented.
- The verification of MAHIS in the dynamic platform construction has been conducted by a case study: PAHIS (Li, Zhang et al. 2004). PAHIS supports the dynamic addition and removal of group-based agents at run-time. A self-organising ring-based architectural model has been proposed to organise the middle agents in PAHIS (Li, Zhang et al. 2003e). Moreover, the ring-based architectural model has been evaluated from the complexity, efficiency, extendibility, and availability points of view (Li, Zhang et al. 2003a).
- The verification of MAHIS in HIS construction has been conducted by two successful case studies: "financial investment planning system" and "petroleum reservoir characterisation system". The former has verified MAHIS in constructing agent-based HIS with tight-coupling hybrid strategy. The latter has verified MAHIS in constructing agent-based HIS with loose-coupling hybrid strategy. PAHIS has been employed in these two systems.
- The evaluation of MAHIS with the framework proposed by Cernuzzi and Rossi (Cernuzzi and Rossi 2002) has been conducted by comparing it with Gaia and MAS-CommonKADS. The *reorganisation attributes* in the evaluation framework have been added according to the characteristics of HIS. The evaluation results indicated that MAHIS is preferable over Gaia and MAS-CommonKADS, especially in the construction of agent-based HIS.

1.6 Outline of the Thesis

The rest of the thesis consists of eight chapters. Chapter 2 is the literature review and related work. The agent-based HIS, the agent-oriented methodologies and their evaluations have been reviewed in this chapter. Chapter 3 to Chapter 8 in the thesis can be divided into two levels. The lower level is the foundation of this thesis and the work toward the proposal of MAHIS methodology, which contains Chapters 3 and 4. Chapter 3 addresses the merits and limitations of the existing well-known agent-oriented methodologies for HIS construction by means of evaluation based on *attributes tree*. Six agent-oriented methodologies have been ranked according to the attributes of each hybrid strategy. MAS-CommonKADS has been selected to be tailored for constructing MAHIS. Chapter 4 introduces MAHIS by tailoring MAS-CommonKADS according to the characteristics of HIS. The upper level is the work toward the competency test of MAHIS, which contains Chapters 5, 6, 7 and 8. Chapter 5 discusses the verification of MAHIS in dynamic platform construction by developing the platform PAHIS following MAHIS. Chapter 6 presents the verification of MAHIS in constructing agent-based HIS with tight-coupling hybrid strategy by developing a case study "financial investment planning system". Chapter 7 presents the verification of MAHIS in constructing agent-based HIS with loose-coupling hybrid strategy by developing a case study "petroleum reservoir characterisation system". Chapter 8 describes the evaluation of MAHIS by comparing MAHIS with Gaia and MAS-CommonKADS based on the *attributes tree*. Finally, Chapter 9 summarises this thesis and put forward some thinking about the future work.

Chapter 2

2 Literature Review and Related Work

HIS is essential for complex problem solving. However, the design and development of these systems are difficult because they have a large number of components with many interactions. Some researchers in the agent research community have produced a qualitative analysis to provide the intellectual justification of precisely why agent-based systems are well suited to engineering HIS. At the same time, some researchers have attempted to use agent perspectives to build their HIS. These studies in agent-based HIS motivate us to conduct formalising agent-based HIS construction. The HIS, suitability of agent perspectives in HIS, and some agent-based HIS are outlined in Section 2.1.

The agent perspective offers a powerful repertoire of tools, techniques, and metaphors that have the potential to considerably improve the way in which people conceptualise and implement many types of software. However, the development of complex multi-agent systems requires not only new models and technologies, but also new methodologies to support developers in an engineered approach to the analysis and design of such systems. During the last decade, more than two dozens of agent-oriented methodologies for developing agent-based systems have been

developed (Sturm and Shehory 2003) although only a few of these methodologies are complete and well-grounded (Zhang 2001, Cuesta, Gomez et al. 2004). These research results are the foundation of our research because we can borrow their successful ideas and avoid building a new methodology from scratch. The current practices of agent-oriented methodologies and their development approaches are discussed in Section 2.2.

After a new methodology is proposed, its competency and suitability should be validated. Methodology evaluation carries out this task. Moreover, once a designer has made the decision to use a multi-agent design, he/she must select the particular methodology that is best suited for the problem he/she is solving. How to choose an appropriate methodology when faced with a set of variable software engineering methodology alternatives is another task of the methodology evaluation. We will employ a practical evaluation approach with enrichment of agent-based HIS construction to validate our proposed methodology. The evaluation approaches to agent-oriented methodologies are discussed in Section 2.3.

2.1 Agent-Based HIS

Agent techniques represent an exciting new means of analysing, designing and building complex software systems. HIS is complex software systems and has all the features of other industrial-strength software systems. Thus, agent-oriented approaches can significantly enhance the ability to model, design and build HIS.

2.1.1 HIS and Hybrid Modelling

In the past decade, the amount of research and development involving HIS has increased rapidly. The integration of expert systems, neural networks, fuzzy systems, and genetic algorithms has proven to be a useful way to develop real-world applications, including the areas of control systems, robotics, diagnostic systems, financial planning, and industrial operations (Medsker 1995, Lertpalangsunti and

Chan 1998, Balducelli and D'Esposito 2000, Li 2000, Yang, Yen et al. 2000, Barhen 2002, Ao 2003, Li, Liu et al. 2004, Zhang and Zhang 2004).

In the hybrid intelligent system community, research work mainly falls into areas like micro-level integration and hybrid integration strategies. The micro-level integration is also called hybrid technique models including fuzzy logic and expert systems, fuzzy systems and neural networks, genetic algorithms and neural networks, genetic algorithms and fuzzy systems, genetic algorithms and expert systems (Zhang and Zhang 2004). Hybrid integration strategy focuses on different levels of intelligent activities and systems, and the individual technologies and their hybrids. Issues range from the fundamental questions about the nature of cognition and theories of computation to problems of exactly how best to implement hybrid systems (Jain and Jain 1997).

HIS can be classified into different categories based on different criteria. Medsker and Bailey discussed the interaction of expert systems and neural networks based on the hybrid integration strategy (Medsker and Bailey 1992, Medsker 1995). Five different hybrid development strategies, namely, *stand-alone*, *transformations*, *loose-coupling*, *tight-coupling*, and *fully-integration*, have been identified. Khosla and Dillon grouped HIS into four classes: fusion systems, transformation systems, combination systems, and associative systems (Khosla and Dillon 1997). However, Medsker and Bailey's approach is more suitable to integrate intelligent techniques in system-level; nevertheless Khosla and Dillon's approach is better to integrate intelligent techniques in micro-level. Their relationships are modelled in details in Chapter 3.

Zhang and Zhang (Zhang and Zhang 2004) have summarised the regularities of the complexity of HIS based on the Simon's discussion (Simon 1996). Firstly, complexity of HIS frequently takes the form of a hierarchy. That is, the system is composed of inter-related subsystems, each of which is in turn hierarchical in structure. Secondly, the choice of the components which are primitive is relatively arbitrary and is defined by the observer's aims and objectives. Thirdly, hierarchical systems evolve more quickly than non-hierarchical ones of comparable size. Finally, it is possible to distinguish between the interactions among subsystems and the

interactions within subsystems. The latter interactions are both more frequent and more predictable than the former ones. Given these observations, software engineers have devised a number of powerful tools in order to tackle this complexity. The principal mechanisms include decomposition, abstraction, and organisation (Booch 1994). Any approach to building HIS should support these three mechanisms (Zhang and Zhang 2004).

2.1.2 Suitability of Agent in HIS

Agents (adaptive or intelligent agents and multi-agent systems) constitute one of the most prominent and attractive technologies in computer science in the last century (Russell and Norvig 1995). Agent and multi-agent system technologies, methods, and theories are currently being contributed to many diverse domains. Agent is not only a very promising technology, it is emerging as a new way of thinking, a conceptual paradigm for analysing problems and for designing systems, for dealing with complexity, distribution and interactive, and perhaps a new perspective on computing and intelligence (Wooldridge 1998).

Unfortunately, there is no universally accepted definition of the term agent, and indeed there is much ongoing debate and controversy on this very subject (Wooldridge and Jennings 1999, Wooldridge and Ciancarini 2001). The definition presented here is adapted from Wooldridge and Jennings (Wooldridge and Jennings 1995).

> *An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.*

Software environments include operating systems, computer applications, databases, networks, and virtual domains. The environment properties are grouped as accessible versus inaccessible, deterministic versus non-deterministic, static versus dynamic, discrete versus continuous (Russell and Norvig 1995).

Agents have the capabilities of being reactive, proactive, and social (Wooldridge and Jennings 1995, Wooldridge 1999). Agents possess the characteristics of *delegacy*, *competency*, and *amenability* (Jennings and Wooldridge 1996). The *delegacy* means discretionary authority to autonomously act on behalf of the client. The *competency* means the capability to effectively manipulate the problem domain environment to accomplish the prerequisite tasks. The *amenability* means the ability to adapt behaviour to optimise performance in an often non-stationary environment in responsive pursuit of the goals of the client. The *amenability* may be combined with accountability.

From a multi-agent perspective, agents in multi-agent systems are autonomous and can engage in flexible, high-level interactions (Jennings and Wooldridge 1996). Here, autonomy means that the agents have their own persistent thread of control and that they can decide for themselves which actions they should perform at what time. The fact that agents are active means they know for themselves when they should be acting and when they should update their state. The flexible nature of interactions means that agents can make decisions about the nature and scope of interactions at run-time rather than design time (Zhang 2001).

Multi-agent systems are ideally suited to representing problems that have multiple problem solving methods, multiple perspectives and/or multiple problem solving entities. Such systems have the traditional advantage of distributed and concurrent problem solving, but have the additional advantage of sophisticated patterns of interactions (Zhang and Zhang 2004). Examples of common types of interaction include cooperation, coordination, and negotiation. It is the flexibility and high-level nature of these interactions that distinguishes multi-agent systems from other forms of software, and provides the underlying power of the paradigm.

Furthermore, Jennings defined the canonical view of a complex system and a multi-agent system (Jennings 2000, Jennings 2001). In the canonical view of a complex system, the system’s hierarchical nature is expressed through the "related to" links, where components within a subsystem are connected through "frequent interaction" links, and interactions between components are expressed through "infrequent interaction" links (Zhang and Zhang 2004) .

From the canonical view of a multi-agent system, it can be seen that adopting an agent-oriented approach to software engineering means decomposing the problem into multiple, interacting, autonomous components that have particular objectives to achieve. The key abstraction models that define the "agent-oriented mind-set" are agents, interactions, and organisations. Finally, explicit structures and mechanisms are often available for describing and managing the complex and changing web of organisational relationships that exist between the agents.

Some researchers in this field have given a qualitative analysis to provide the intellectual justification of precisely why agent-based systems are well suited to engineering complex software systems (Jennings 2000, Jennings 2001, Zhang 2001, Zhang and Zhang 2004). They have also provided a detailed analysis of the merits of agent-oriented decomposition, the appropriateness of agent-oriented abstractions, and the need for flexible management of changing organisational structures in the process of building complex software systems. On the other hand, it is evident that HIS is complex software systems and has all the features of other industrial-strength software systems. Thus, agent-oriented approaches can significantly enhance our ability to model, design and build HIS for the following reasons (Zhang 2001, Zhang and Zhang 2004):

- **Merits of agent-oriented decomposition.** HIS consists of a number of related subsystems organised in a hierarchical fashion. At any given level, subsystems work together to achieve the functionality of their parent system. Moreover, within a subsystem, the constituent components work together to deliver overall functionality. Thus, the same basic model of interesting components, working together to achieve particular objectives, occurs throughout the system. The agent-oriented approach advocates decomposing problems in terms of autonomous agents that can engage in flexible, high-level interactions. The fact that agents are active means they know for themselves when they should be acting and when they should update their state. Such self-awareness reduces control complexity since the system's control know-how is taken from a centralised repository and localised inside each individual problem solving component. The fact that agents make decisions about the nature and scope of

interactions at run-time makes the engineering of HIS easier for two main reasons. Firstly, the system's inherent complexity means that it is impossible to know a priori about all potential links. Interactions will occur at unpredictable times, for unpredictable reasons, and between unpredictable components. For this reason, it is futile to try and predict, or analyse, all the possibilities at design-time. Rather, it is more realistic to endow the components with the ability to make decisions about the nature and scope of their interactions at run-time. Thus agents are specifically designed to deal with unanticipated requests, and they can spontaneously generate requests for assistance whenever appropriate. Secondly, the problem of managing control relationships between the software components is significantly reduced.

- **Suitability of agent-oriented abstractions.** A significant part of the design process is finding the right models for viewing the problem. In general, there will be multiple candidates, and the difficult task is to pick out the most appropriate one. When designing software, the most powerful abstractions are those that minimise the semantic gap between the units of analysis that are intuitively used to conceptualise the problem and the constructs present in the solution paradigm. In the case of complex HIS, the problem to be characterised consists of subsystems, subsystem components, interactions and organisational relationships. Taking each in turn: subsystems naturally correspond to agent organisation; the appropriateness of viewing subsystem components as agents has been made above; the interplay between the subsystems and between their constituent components is most naturally viewed in terms of high-level social interactions; and complex HIS involves changing webs of relationships between their various components. They also require collections of components to be treated as a single conceptual unit when viewed from a different level of abstraction. Here again the agent-oriented mind-set provides suitable abstractions.
- **The need for flexible management of changing organisational structures.** Organisational constructs are first-class entities in agent systems. Thus, explicit representations are made of organisational relationships and structures.

> Moreover, agent-based systems have the concomitant computational mechanisms for flexible forming, maintaining and disbanding organisations. This representational power enables agent-oriented systems to exploit two facets of complex HIS. Firstly, the notion of a primitive component can be varied according to the needs of the observer. Thus at one level, entire subsystems can be viewed as a singleton, a collection of agents can be viewed as primitive components, and so on, until the system eventually bottoms out. Secondly, such structures provide a variety of stable intermediate forms. These forms are essential for rapid development of complex HIS. Their availability means that individual agents or organisational groupings can be developed in relative isolation, and then added into the system in an incremental manner. This, in turn, ensures that there is a smooth growth in functionality.

Therefore, it is apparent that multi-agent perspectives are well suitable for modelling HIS when solving complex problems. Agent-based systems are highlighted as promising tools for complex problems with many changes and disturbances at run-time (Monostori 2003).

2.1.3 Agent-Based Hybrid Intelligent Systems

In recent years, many researchers and practitioners have been engaging in the area of agent-based HIS. Their research focuses on the agent-based hybrid intelligent applications and agent-oriented models for developing agent-based HIS.

One of the main tasks of agent-based hybrid intelligent application development is to incorporate intelligent techniques into individual agents. There are three principal methods to incorporate intelligent techniques into individual agents from implementation point of view: *via* .DLL (dynamically linked libraries) or other callable APIs (application programming interface), through specific interface agents and stand-alone intelligent systems, and intelligent technique components as OO class (Zhang 2001). As the practices of the above methods, Sobh and Bajcsy implemented a discrete event dynamic system observer for a moving agent (Sobh

and Bajcsy 1992). The agent can manipulate an object with hybrid intelligent mechanism. Zha, Lim et al. designed a unified class of Petri nets OOIPNs based on intelligent agents and HIS (Zha, Lim et al. 1997). The hybrid and dynamic knowledge can be exchanged among intelligent agents. Lauber, Steger et al. presented a hybrid technique for autonomous diagnosis agents in combining quality assurance data, failure mode and effects analysis and probabilistic causal reasoning (Lauber, Steger et al. 1999). Marwaha, Chen et al. proposed a routing scheme with hybrid technique for mobile ad hoc networks (Marwaha, Chen et al. 2002). The scheme used mobile agents to combine the individual intelligent techniques. Fregene, Madhavan et al. proposed a pursuit-evasion scheme involving multiple coordinated vehicle teams in which each vehicle was modelled as a class of hybrid intelligent agents (Fregene, Kennedy et al. 2003, Fregene, Madhavan et al. 2004). Yang, Yen et al. developed an intelligent personal spider agent which was based on automatic textual analysis of the Internet documents and hybrid simulated annealing (Yang, Yen et al. 2000).

Another task of agent-based hybrid intelligent application development is the construction of framework and architecture. The framework and architecture integrate two or more intelligent techniques with multiple agents. Thus far, there is some research involved in this topic. One of such attempts is the MIX multi-agent platform (Iglesias, Centeno-Gonzalez et al. 1995). The focus of the MIX is the development of strategies and tools for integrating neural and symbolic techniques.

Other such attempts are summarised as following. Averbukh developed a hybrid intelligent architecture intended for work in complex real time client-server environment (Averbukh 1999). The architecture provides integration of imperative high-level actions of an intelligent agent with work of active elements possessing own behaviour and based on recurrent neural networks schemes. Zhang, Li et al. developed a hybrid intelligent and mobile agent platform (Zhang, Li et al. 2003). The platform employed the underlying derivation/adduction and mobility mechanism. The agents cooperated to perform a task under a uniform platform. Kim, Heragu et al. presented a hybrid intelligent agent-based scheduling and control system architecture for an actual industrial warehouse order-picking problem (Kim,

Heragu et al. 2003). The architecture included a higher level optimizer, a middle-level guide agent, and lower level agents. A mathematical model and a genetic algorithm were designed. Varga, Jennings et al. integrated intelligent systems into an agent cooperating community for electricity distribution management based on ARCHON architecture (Varga, Jennings et al. 1994). Van Breemen and de Vries developed an agent-based framework which enabled the use of heterogeneous control algorithms and integrating intelligent techniques (van Breemen and de Vries 2001). Balducelli and D'Esposito designed a multi-agent architecture for emergency management problem (Balducelli and D'Esposito 2000). The architecture included case-based reasoning (CBR) agents, simulation agents, and genetic agents. Feijo and Bento presented a programming environment to support the development of CAD systems based on a hybrid agent architecture in which the symbolic reasoning war carried out by first-order logic (Bento and Feijo 1997, Feijo and Bento 1998). Brussel, Wyns et al. presented a holonic reference architecture for manufacturing systems (Brussel, Wyns et al. 1998). Staff holons could be added to assist the basic holons with expert knowledge. Hu and Brady proposed a parallel processing architecture used for real-time, sensor-based control of mobile agents (Hu and Brady 1996). This architecture carried out hybrid control based on locally intelligent control agent. Velasco, Gonzalez et al. designed an architecture and a platform to implement distributed applications as a set of cooperating intelligent agent (Velasco, Gonzalez et al. 1996). Li, Liu et al. developed a multi-agent framework and a ring-based architectural model to organize components and interactions. The framework consists of user interface, decision making, knowledge discovering, information management facilitators, and distributed heterogeneous data resources (Li, Liu et al. 2004). Zhang and Zhang proposed an approach to constructing agent-based HIS using middle agent technique (Zhang and Zhang 2000a).

The approaches used in the above systems have the following limitations.

- It is impossible to embed many intelligent techniques with a single agent. Otherwise, the agents will be overloaded. In many applications, the agents

should be kept simple for ease of maintenance, initialisation, and customisation.

- It is not flexible to add more intelligent techniques to or remove some unwanted ones from the multi-agent systems.
- The agents in these systems are difficult to inter-operate as they did not use some type of common or standard agent communication language.
- These frameworks, architecture, and systems were usually developed from scratch rather than by following a methodology. The reason is that no one methodology can fully manage the analysis and design of the agent-based HIS (Zhang and Zhang 2004).

However, some models or methodologies for developing agent-based HIS have been proposed. Srikanth proposed an architecture for a multi-agent hybrid intelligent system that can trade in multiple markets (Srikanth 1999). The contribution of this research is that the concept of methodology was taken into account. The process stages were divided into requirements identification, analysis, and design. Some results were presented in UML notation. Khosla and Dillon introduced a computational architecture called IMAHDA (Khosla and Dillon 1997). The role and knowledge content of IMAHDA consisted of four layers: object, software agent, intelligent agent, and problem solving agent. The IMAHDA can be seen as being constructed from generic software agents (distributed processing, distributed communication, belief base, and relational software agents), generic intelligent agents (expert/knowledge based system, supervised neural network, unsupervised neural network, fuzzy logic, genetic algorithm intelligent agents), and problem solving agents (global pre-processing, decomposition, control, decision, and post processing agents). Zhang and Zhang proposed a methodology for HIS construction by combining Gaia (Wooldridge, Jennings et al. 2000), the coordination-oriented methodology (Zambonelli, Jennings et al. 2001) and the organisation abstraction (Zambonelli, Jennings et al. 2000, Zhang and Zhang 2004). The methodology consisted of six models: agent model, role model, skill model, knowledge model, organisation model, and interaction model. The construction of these models was divided into five steps: identification of the roles, identification of

skills, modelling of the knowledge, design of organisation structure, and analysis of dynamics.

These methods or methodologies are not enough to cover the entire characteristics of HIS. At the same time, they have the following limitations: lack of formal description, the loose linkage between the outputs of components, and inflexible mechanism to construct agent-based HIS.

2.2 Methodologies for Multi-Agent Systems

Agent technology has received a great deal of attention in the last few years and, as a result, the industry is beginning to get interested in using this technology to develop its own products. In spite of the different developed agent theories, languages, architectures and successful agent-based applications, some works for specifying techniques to develop applications using agent technology have been done. The role of agent-oriented methodologies is to assist in all the phases of the life cycle of an agent-based application, include its management. This section provides a brief overview of the state of the art in the area of software engineering methodologies for multi-agent systems.

2.2.1 Concept of Agent-Oriented Methodology

A methodology is the set of guidelines for covering the whole lifecycle of system development both technically and managerially (Graham, Henderson-Sellers et al. 1997). It must provide a set of concepts, the usage rules of these concepts by organising them into various steps, the process associated with these steps and a notion (Gervais 2003), which should concretely include the following: a full lifecycle process; a comprehensive set of concepts and models; a full set of techniques (rules, guidelines, heuristics); a fully delineated set of deliverables; a modelling language; a set of metrics; quality assurance; coding (and other) standards; reuse advice; and guidelines for project management (Graham,

Henderson-Sellers et al. 1997). It then provides a guideline to software engineers to explain how they can describe the architecture of the system they are building, from the most abstract level to the most concrete level, which is the implementation of the system. The relationships between these components are shown in Figure 2.1 (Graham, Henderson-Sellers et al. 1997).

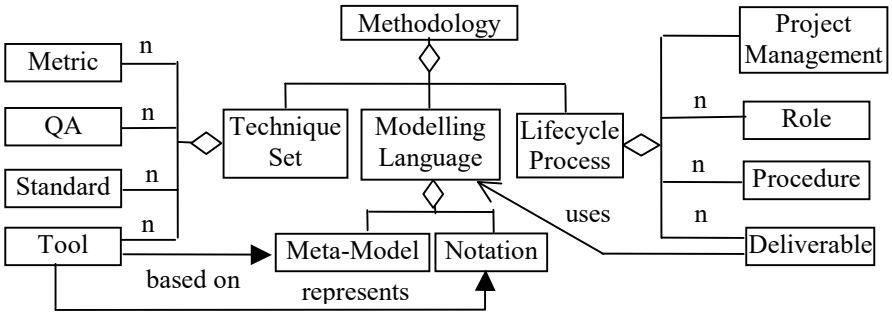

Figure 2.1 The relationships among components of a methodology

The primary objective of a methodology is that of constructing valid systems of a given meta-model. First of all, the construction process (called methodological process) must allow the constructing of valid systems. However, this process is based on a meta-model of the constructed systems that must also be valid. The classical methodologies are based on well-founded meta-models and theories. A methodology will not be precise, clear and complete until its meta-model is so (Gervais 2003).

The agent-oriented methodology is to assist an agent-based application in all of its lifecycle phases. It offers a new perspective to developing MAS by increasing the level of abstraction the developer users to analyse and design the system (O'Malley and Deloach 2001). It enables inexpensive development and maintenance of MAS which should be flexible, easy-to-use, and scalable and of high quality. The first goal of a multi-agent methodology is to allow the construction of MAS. Some methodologies are modifications of object-oriented techniques without taking into accounts the first class attributes of multi-agent systems. The argumentation for such works is the simplification of the development process and its acceptance by developer, in large and long experiments. Where simplicity and experimentation are

important factors for a methodology, these factors are not sufficient to accept it as a multi-agent methodology. A methodology that does not build MAS could not be considered as an agent-oriented methodology regardless of its possible simplicity and/or familiarity.

2.2.2 Classification of Agent-Oriented Methodologies

To avoid building a methodology from scratch, the researchers on agent-oriented methodologies have followed the approach of existing methodologies to include the relevant aspects of the agents. In the survey of agent-oriented methodologies by Iglesias et al. (Iglesias, Garijo et al. 1998), the methodologies are classified into those extending object-oriented methodologies and those extending knowledge engineering methodologies (Bienvenido and Flores-Parra 2003, del Aguila, Canadas et al. 2003). However, traditional methodologies for analysis and design are poorly suited to multi-agent systems because of the fundamental mismatch between the respective levels of abstraction (Zhang 2001). Despite this mismatch, several proposals do take object-oriented modelling techniques or methodologies as their basis. On the one hand, some proposals directly extend the applicability of object-oriented methodologies and techniques to the design of agent systems (Jennings, Sycara et al. 1998, Huget 2002). However, these proposals fail to capture the autonomous and proactive behaviour of agents, as well as the richness of their interactions. On the other hand, some proposals seek to extend and adapt object-oriented models and techniques to define a methodology for use in multi-agent systems (Kinny, Georgeff et al. 1996, Kinny and Georgeff 1997, Georgeff, Pell et al. 1999). Although these proposals can sometimes achieve a good modelling of the autonomous behaviour of agents and of their interactions, they lack the conceptual mechanisms for adequately dealing with organisations and agent societies.

A different set of proposals build upon, and extend, methodologies and modelling techniques from knowledge engineering. These techniques provide formal and compositional modelling languages for the verification of system

structure and function. These approaches are well-suited to modelling knowledge-oriented and information-oriented agents (Brazier, Dunin-keplicz et al. 1997, Klein 2003). However, since these approaches usually assume a centralised view of knowledge-based systems, they fail to provide adequate models and support for the dynamic view of multi-agent systems.

This classification, followed by the majority of the researchers of this domain, is based on the implementation of MAS. Generally, this implementation is achieved, either in the form of an object-oriented system or expert system. However, several methodologies such as the organisational ones cannot be classified according to this classification. Lahlouhi and Sahnoun proposed another classification that is based on the MAS development process (Lahlouhi and Sahnoun 2002). In this approach, the methodologies can be classified as follows: complementary methodologies, modelling methodologies, organisational methodologies, and complete methodologies.

The complementary methodologies are those allowing the implementation of MAS models. They are interested in the implementation of the multi-agent models, generally, in the form of object-oriented systems. This class includes all works on the adaptation of the object-oriented model to support agents. The representative works of this class are those of Odell and Parunak on UML extension, MESSAGE/UML, MESSAGE. Several other works can be put in this class.

The modelling methodologies are those that are interested only in the multi-agent models. They consider the aspects concerning the derivation of the multi-agent models starting from requirements without considering their implementation. Several works can be put in this class, such as MAS-CommonCADS, ODAC, MaSE, Tropos, Prometheus, ADELFE, SODA and Zeus.

The organisational methodologies are those that are interested in the organisations to use in a multi-agent model. They allow deriving the multi-agent model starting from the organisation. This class can be subdivided in three subclasses. The subdivision of this is as follow:

- Bottom-up methodologies: These methodologies start with the agents' development, or re-use already developed agents, and then they assign them organisation roles. Cassiopeia and Tropos can be put in this subclass.
- Top-down methodologies: These methodologies begin with the organisational aspects identification and then the agents' development that can accomplish them. ALAADDIN, MaSE, Gaia and its extension, and Styx can be put in this subclass.
- Mixed methodologies: These methodologies begin, indifferently, from the development of the organisation or some agents, assign roles to these agents and then develop agents for the remaining roles. The new version of MASA-Method can be classified in this subclass.

The complete methodologies are those where the methodological process involves, at least, all phases of the minimal organisational process. The multi-agent methodological process must include the following phases: firstly, description of the organisational aspects; secondly, description of a multi-agent system's model that imitate the organisation; lastly, implementation or simulation of the multi-agent model.

The common disadvantage, to all non-organisational methodologies, is that they do not take into account the organisational aspects in the development of MAS which are actually fundamental. This importance is shown in the current tendency of the researchers in multi-agent methodologies, which try to integrate organisational aspects in their methodologies. However, for the moment, few methodologies can be qualified of organisational (Lahlouhi and Sahnoun 2002).

2.2.3 Well-Known Agent-Oriented Methodologies

MAS can be mainly grouped into two classes: a) *distributed problem solving systems* in which the component agents are explicitly designed to cooperatively achieve a given goal, and b) *open systems* in which agents, not necessarily co-designed to share a common goal, can dynamically leave and enter the system

(Zambonelli, Jennings et al. 2000). In the former case, there are some well-known methodologies, for example, Gaia (Wooldridge, Jennings et al. 2000, Zambonelli, Jennings et al. 2003), MAS-CommonKADS (Iglesias, Garijo et al. 1996), MaSE (Deloach, Wood et al. 2001), ODAC (Gervais 2003), Prometheus (Padgham and Winikoff 2002, Padgham and Winikoff 2003), and Tropos (Bresciani and Giorgini 2002, Mouratidis, Giorgini et al. 2002, Giunchiglia, Mylopoulos et al. 2003), for the analysis and design of the first class MAS. In the latter case, the dynamic arrival of unknown agents needs to be taken into account, as well as the possibility of self-interested behaviour in the course of the interactions (Wood and Deloach 2001). This is the organisational aspect of agent-based system. Some attempts have been made in this case (Sierra, Faratin et al. 1997, Cabri, Leonardi et al. 2002, Goldman and Rosenschein 2002, Cao, Li et al. 2003, Ferber, Gutknecht et al. 2003, Graham, Decker et al. 2003, Jouvin and Hassas 2003, Juan, Stering et al. 2003, Kaminka, Frank et al. 2003, Klostermeyer and Klemm 2003, Liu, Sun et al. 2003, Norman, Preece et al. 2003, Yan, Mao et al. 2003, Odell, Parunak et al. 2003a, Odell, Parunak et al. 2003b, Huynh, Jennings et al. 2004, Li, Zhang et al. 2004, Nair, Tambe et al. 2004). However, most well-ground methodologies do not support the organisational characteristic of MAS and they are only suitable to model *close systems* (Cuesta, Gomez et al. 2004).

There are more than two dozens agent-oriented methodologies nowadays (Sturm and Shehory 2003). Even though many agent-oriented methodologies have been proposed, few are mature or described in sufficient detail to be of real use (Dam and Winikoff 2003). However, some agent-oriented methodologies are well-known. These well-known methodologies were selected since they (Dam and Winikoff 2003): (a) were described in more detail (e.g. journal paper rather than a conference paper); and (b) were perceived as being significant by the agent community. Another important factor was whether the methodology had been developed over an extended time period based on feedback from users other than the developers of the methodology. Six methodologies: Gaia, MAS-CommonKADS, MaSE, ODAC, Prometheus, and Tropos were selected based on these criteria and are briefly introduced as following.

- **Gaia methodology**

The Gaia methodology represents one of the few attempts specifically tailored to the analysis and design of MAS, and which deals with both the micro (intra-agent) level and the macro (inter-agent) level of analysis and design. Gaia makes explicit use of an organisational point of view. In this methodology, analysis and design are well-separated phases. The analysis aims to develop an understanding of the system and its structure, in terms of the roles that have to be played in agent organisation and interaction, without any reference to implementation details. The design phase aims to define the actual structure of the agent system of the services to be provided by each agent, and of the acquaintances' structure. However, Gaia, as it presently stands, is not a general methodology for all kinds of MAS. Rather, it is intended to support the development of distributed problem solvers in which the system's constituent components are known at design time (i.e. it is a closed system) and in which all agents are expected to cooperate toward the achievement of a global goal. For these reasons, Gaia is not suitable for modelling open systems, and for controlling the behaviour of self-interested agents.

- **MAS-CommonKADS methodology**

MAS-CommonKADS extended CommonKADS methodology (Schreiber, Wielinga et al. 1994, Sandberg and de Hoog 1996) for MAS modelling, adding techniques from object-oriented methodologies and from protocol engineering for describing the agent protocols. The overall MAS-CommonKADS methodology for MAS developments follows six phases: conceptualisation, analysis, design, coding and testing of each agent, integration, and operation and maintenance (Iglesias, Garijo et al. 1997). MAS-CommonKADS methodology consists of the development of seven models: *Agent Model*, which describes the characteristics of each agent; *Task Model*, which describes the tasks that the agents carry out; *Expertise Model*, which describes the knowledge needed by the agents to achieve their goals; *Organisation Model*, which describes the structural relationships between agents (software agents and/or human agents); *Coordination Model*, which describes the dynamic relationships between software agents; *Communication Model*, which describes the dynamic relationships between human agents and their respective

personal assistant software agent; and *Design Model*, which refines the previous models and determines the most suitable agent architecture for each agent, and the requirements of the agent network. This methodology has been applied or extended in different areas (Sandberg and de Hoog 1996, Post, Wielinga et al. 1997, Kingston 1998, Allsopp, Harrison et al. 2002, Medina, Sanchez et al. 2004).

- **MaSE methodology and its extension**

The primary focus of MaSE is to help a designer take an initial set of requirements and to analyse, design, and implementation a working multi-agent system (Deloach, Wood et al. 2001). The MaSE methodology is a specialisation of more traditional software engineering methodologies. The general operation of MaSE follows analysis and design phases. The purpose of the MaSE analysis phase is to produce a set of roles whose tasks describe what the system has to do to meet its overall requirements. The MaSE analysis phase consists of three steps: Capturing goals, Applying Use Cases, and Refining Roles. The design phase has four steps: Creating Agent Classes, Constructing Conversations, Assembling Agent Classes, and System Design. A major strength of MaSE is the ability to track changes throughout the process. Every object created during the analysis and design phases can be traced forward or backward through the different steps to other related objects.

The extension of MaSE consists of meta-models and a detailed global methodological process (Lahlouhi and Sahnoun 2002). The great reproach, which one can make to MaSE methodology, is that it is too much object-oriented. It uses the principal diagrams of object-oriented methodologies and, in particular, those of the UML meta-model. The introduction of the system's objectives into the analysis phase makes it possible to take into account the co-operation in a more coherent way, such as it is expressed by the system requirements. However, the relation between this description and the other parts of the system, in particular other diagrams and agents, is no clear. This description is in the form of "and/or" hierarchy, which means that the achievement of an objective is done by the achievement of all its sub-objectives (relation "and") or of at least one (relation "or").

It would be more important if this was done at the organisational level not at the multi-agent level. The objective diagrams do not describe coordination between the achievements of sub-objectives. This is necessary to do it on the organisational level.

- **ODAC methodology**

The ODAC methodology (Open Distributed Applications Construction) aims to provide a designer of agent-based systems with a set of methods and tools to allow him/her to control the construction process complexity of such systems (Gervais 2003). It identifies the steps that the methodology defines. The ODAS methodology consists of the following steps: requirement analysis, system analysis and design and finally the implementation in a target environment reflecting as well as possible the real environment in order to carry out the tests. The analysis relates to the development of a detailed solution that does not depend on achievement choices. It includes the description of all the processes composing the system operation, the information definition and tasks' specification. The purpose of the design is to lead to the system implementation and to carry out choices to make the models operational for the system achievement. It relates to the operational specifications needed to ensure the achievement of the future system. It includes the taking into account of the means chosen for adopted solution. Two categories of specifications are identified in ODAC: behavioural specification of the system and its operational specification. The behavioural specification is the output of the analysis. It describes the system according to its objective, its place in the company in which it is developed, information that it handles and the tasks that it carries out. The operational specification results from the projection of the behavioural specification on a target environment reflecting the real executing environment. It constitutes the description from which code is generated and the implementation is carried out.

- **Prometheus methodology**

Another agent-oriented methodology that has proven effective in assisting developers to design, document, and build agent systems is the Prometheus methodology (Padgham and Winikoff 2003). This methodology supports, in particular, the development of BDI-like agents.

The Prometheus methodology consists of three phases. The system specification phase focuses on identifying the basic functionalities of the system, along with inputs (percepts), outputs (actions) and any important shared data sources. The architectural design phase uses outputs from the previous phase to determine which agents the system will contain and how they will interact. The detailed design phase looks at the internals of each agent and how it will accomplish its tasks within the overall system.

Prometheus differs from other existing methodologies in that it (Padgham and Winikoff 2003):

- ✓ Supports the development of intelligent agents which use goals, beliefs, plans, and events. By contrast, many other methodologies treat as "simple software processes that interact with each other to meet an overall systems goal".
- ✓ Provides "start-to-end" support (from specification to detailed design and implementation) and a detailed process, along with design artefacts constructed and steps for deriving artefacts.
- ✓ Provides hierarchical structuring mechanisms which allow design to be performed at multiple levels of abstraction.
- ✓ Uses an iterative process over software engineering phases rather than a linear "waterfall" model. Although the phases are described in a sequential fashion, the intention is not to perform them purely in sequence.
- ✓ Provides cross checking of design artefacts.

However, Prometheus has its own limitations. Because Prometheus specially supports the development of BDI-like agents, this makes Prometheus less useful to those developing non-BDI-like agents.

- **Tropos methodology**

Tropos is an agent-oriented software development methodology founded on two key features: a) the notions of agent, goal, plan and various other knowledge level concepts are fundamental primitives used uniformly throughout the software development process; and b) a crucial role is assigned to requirements analysis and specification when the system-to-be is analysed with respect to its intended

environment. The Tropos methodology includes five phases, namely, *Early Requirements, Late Requirements, Architectural Design, Detailed Design, and Implementation* (Giunchiglia, Mylopoulos et al. 2003). During *Early Requirements* phase the relevant stakeholders are identified, along with their respective objects; stakeholders are represented as actors, while their objectives are represented as goals. During *Late Requirements* the system-to-be is introduced as another actor and is related to stakeholder actors in terms of actor dependencies; these indicate the obligations of the system toward its environment, also what the system can expect from actors in its environment. During *Architectural Design* phase more system actors are introduced and they assigned subgoals or subtasks of the goals and tasks assigned to the system. During *Detailed Design* phase system actors are defined in further detail, including specifications of communication and coordination protocols. During *Implementation* phase, the Tropos specification, produced during detailed design, is transformed into a skeleton for the implementation. This is done through a mapping from the Tropos constructs to those of an agent programming platform; code is added to the skeleton using the programming language supported by the programming platform.

The Tropos conceptual models and diagrams are developed as instances of following intentional and social concepts: actor, goal, dependency, plan, resource, capability, and belief. The notion of actor models an entity that has strategic goals and intentionality. An actor represents a physical agent, or a software agent as well as a role or a position. A role is an abstract characterisation of the behaviour of an actor within some specialised context, while a position represents a set of roles, typically played by one agent. An agent can occupy a position, while a position is said to cover a role. Notice that the notion of actor in Tropos is a generalisation of the classical AI notion of software agent. A goal represents the strategic interests of actors. The framework distinguishes between hard goals and soft goals, the latter having no clear-cut definition and/or criteria as to whether they are satisfied. Soft goals are useful for modelling software qualities, such as security, performance and maintainability. A dependency between two actors indicates that one actor depends on another in order to attain some goal, execute some plan, or deliver a resource. A

plan represents a way of satisfying a goal. A resource represents a physical or an informational entity that one actor wants and another can deliver. A capability represents the ability of an actor to define, choose and execute a plan to fulfil a goal, given a particular operating environment. Beliefs are used to represent each actor's knowledge of the world.

2.3 Agent-Oriented Methodology Evaluations

Some evaluation frameworks have been proposed to compare agent-oriented methodologies (O'Malley and Deloach 2001, Shehory and Sturm 2001, Cernuzzi and Rossi 2002, Dam and Winikoff 2003, Mali 2003, Sturm and Shehory 2003, Cuesta, Gomez et al. 2004). These approaches found two sources of features to evaluate the methodologies. Firstly, they adopt general software-engineering criteria, which have been found relevant for evaluations of methodologies according to various paradigms. Secondly, they identified specific criteria that are needed to support agent-oriented concepts in development. All of them gather a set of criteria, which is supposed to be independent from the field of application or platform. They point out what is needed for a comprehensive methodology together with individual drawbacks and advantages. However, all these evaluations did not take the characteristics of HIS into account.

Shehory and Sturm (Shehory and Sturm 2001) performed a feature-based evaluation of several agent-oriented methodologies. Their criteria include software engineering related criteria and criteria relating to agent concepts. They used the same techniques in addition to a small experimental evaluation to perform an evaluation of their own Agent Oriented Modelling Techniques (AOMT) (Sturm and Shehory 2003). This work suffers from subjectivity in that the criteria they identified are those that they see as important and, naturally, AOMT focuses on addressing these criteria (Dam and Winikoff 2003).

A framework to carry out an evaluation of agent-oriented analysis and design modelling methods has been proposed by Cernuzzi and Rossi (Cernuzzi and Rossi

2002). The proposal makes use of feature-based evaluation techniques but metrics and quantitative evaluations are also introduced. The significance of the framework is the construction of an attribute tree, where each node of the tree represents a software engineering criterion or a characteristic of agent-based system. Each attribute is assigned with a score and the score of attributes on the node is calculated based on those of their children.

O'Malley and DeLoach (O'Malley and Deloach 2001) proposed a number of criteria for evaluating methodologies with a view to allowing organisations to decide whether to adopt agent-oriented methodologies or use existing OO methodologies. Although they performed a survey to validate their criteria, they do not provide detailed guidelines or a method for assessing methodologies against their criteria. Their work is useful in that it provides a systematic method of taking a set of criteria, weightings for these criteria and an assessment of a number of methodologies and determining an overall ranking and an indication of which criteria are critical to the result (Dam and Winikoff 2003).

2.4 Summary

Agent technique is suitable for constructing HIS. One of the reasons which affect agent techniques to be further used in industrial hybrid intelligent applications, is the lack of methodologies for constructing agent-based HIS. Although many agent-oriented methodologies have been proposed, a few has taken the characteristics of HIS into account. The evaluation to clarify the gap between these existing agent-oriented methodologies and agent-based HIS construction should be conducted.

Some agent-oriented methodology evaluation frameworks have been proposed. However, they cannot be directly employed to evaluate methodologies for agent-based HIS construction because these evaluation frameworks did not consider the characteristics of HIS. In order to propose a methodology for agent-based HIS construction, two major tasks need to be done based on the descriptions in this Chapter.

- Enrichment of a competent evaluation framework with the characteristics of HIS for evaluating the existing well-known methodologies;
- Evaluation of the existing well-known agent-oriented methodologies with the enriched framework for clarifying the gap between the current existing well-known agent-oriented methodologies and agent-based HIS construction.

These tasks will be conducted in next Chapter.

Chapter 3

3 Hybrid Modelling and Agent-Oriented Methodologies

From the discussion in previous chapters, it is no doubt that there is a blooming of hybrid techniques. At the same time, it is concluded that agent perspective is suitable for modelling, designing, and constructing HIS (Jennings 2001, Zhang and Zhang 2004). However, the number of deployed commercial agent-based hybrid intelligent applications is small. One of the key problems is the lack of a competent methodology in agent-based HIS construction. Although many agent-oriented methodologies have been proposed during the last decade, there is no one methodology that has considered all the characteristics of HIS. An agent-oriented methodology for constructing HIS needs to be urgently proposed in the agent-based HIS field. To avoid building the methodology from scratch, it is wise to employ the ideas and components of the existing well-known methodologies. For starting the work, the following questions must be clarified first. What are the distinct characteristics of HIS rather than general complex problems? Can the existing agent-oriented methodologies be ranked according to the characteristics of HIS? What is the gap between the existing agent-oriented methodologies and agent-based HIS construction? This chapter attempts to answer these questions. Section 3.1

introduces the concept of HIS. Section 3.2 discusses hybrid modelling from hybrid integration strategy point of view. The characteristics of HIS are summarised in this section. Section 3.3 presents the gap between the existing well-known agent-oriented methodologies and agent-based HIS construction by means of evaluating and ranking these methodologies with the enriched *attributes tree*.

3.1 What Is A Hybrid Intelligent System?

A system is intelligent if it is able to improve its performance or maintain an acceptable level of performance in the presence of uncertainty. The main attributes of intelligence are learning or generalisation, adaptation, fault tolerance and self repair, and self-organisation (Medsker 1995). The ability of the system to examine and modify its behaviour in a limited sense is usually achieved by using expert systems also called knowledge-based systems, artificial neural networks, fuzzy logic systems, genetic algorithms, case-based reasoning, and so on. The intelligent systems using these techniques have accomplished substantial gains but there are problems associated with them too.

The combined use of these complementary techniques to solve complex problems are becoming popular and thus paving the way for the emergence of HIS. HIS, as depicted in Figure 3.1, is usually implemented using traditional hard computing techniques (e.g., expert systems) and soft computing techniques (e.g., neural networks, fuzzy systems, and genetic algorithms) (Medsker 1995, Zhang and Zhang 2004).

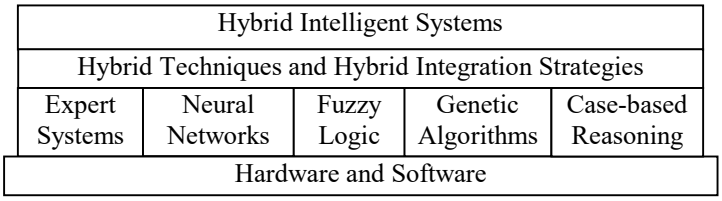

Figure 3.1 Intelligent technologies being used in HIS

The modern trend is to integrate these individual intelligent techniques by using hybrid techniques and hybrid integration strategies (hybrid strategies for short) for offsetting the demerits of one technique by the merits of another technique. The resulting systems are called HIS (hybrid intelligent systems).

Expert systems and neural networks are well established as useful technologies that in fact complement each other in powerful HIS. Many development tools and environments are now coming out specifically for the development of hybrid neural network and expert systems. Other technologies that have more recently been exploited are fuzzy logic, genetic algorithms, case-based reasoning, and so on. Developers are finding niches for each of these and the various combinations of the technologies are being explored and used.

Nowadays, in the HIS community researchers mainly focus on *hybrid techniques*, *hybrid strategies*, and *hybrid modelling*. The *hybrid techniques* are efficient methods to combine two or more individual intelligent techniques. The *hybrid strategy* focuses on different levels of intelligent activities and systems, and the individual technologies and their hybrids. Issues range from the fundamental questions about the nature of cognition and theories of computation to problems of exactly how best to implement hybrid systems. The *hybrid modelling* focuses on the models of hybrid techniques and hybrid strategies for abstracting and formalising the techniques and strategies in hierarchy.

3.2 Hybrid Modelling

Hybrid modelling carries out the construction of the hybrid technique models and hybrid strategy models. Many practical hybrid techniques have been proposed based on three methods, namely, *combination*, *transformation*, and *fusion*. These three methods can be regarded as three relations on the set of intelligent techniques. So a hybrid technique model will combine two or more individual intelligent techniques by using relations: *COMBINATION*, *TRANSFORMATION*, and *FUSION*. Hybrid strategy model mainly focuses on the implementations of HIS in this thesis.

The aim of the hybrid strategy is to integrate different hybrid technique models into a system by using a suitable hybrid strategy model. So far, five hybrid strategy models have been proposed, that is, stand-alone model, transformational model, loose-coupling model, tight-coupling model, and fully-integration model. There are obvious relationships between the hybrid relations and these hybrid strategy models.

3.2.1 Hybrid Technique Models

Many hybrid technique models or architectures have been developed, e.g., hybrid neural network and expert systems, combination of fuzzy logic and expert systems, fuzzy systems with neural networks, genetic algorithms with neural networks, neural networks with genetic algorithms, genetic algorithms with fuzzy systems, combination of genetic algorithms and expert systems, hybrid systems with case-based reasoning (Medsker 1995, Jain and Jain 1997, Lertpalangsunti and Chan 1998, Li 2000). For formally describing the hybrid technique models, a set of intelligent techniques and some relations are defined as following.

Let

$$X = \{ES, NN, FL, GA, CBR, \cdots\}$$

where ES means Expert Systems; NN means Neural Networks; FL means Fuzzy Logic; GA means Genetic Algorithms; CBR means Case-Based Reasoning.

Three relations: *COMBINATION* ($\Vdash$), *TRANSFORMATION* ($\Rightarrow$), and *FUSION* (Θ) are defined on X as shown in Table 3.1. Note: x and y are intelligent techniques and $x, y \in X$.

Table 3.1 Relations used in hybrid technique models

Operators	Interpretation
$x \Vdash y$	Knowledge representation and processing of techniques x and y are interleaved
$x \Rightarrow y$	Knowledge representation of technique x is transformed into the one of technique y
$x \Theta y$	Intelligent technique x and y share the same knowledge representation

COMBINATION ($\Vdash$) involves explicit hybridisation. This relation models the different levels of information processing and intelligence by using intelligent

techniques which best model a particular level. It involves a modular arrangement of two or more intelligent techniques to solve real world problems. $x \Subset y$ denotes that x and y are fully interleaved, whether knowledge representation or processing.

TRANSFORMATION ($\Vdash$) is used to transform one form of representation into another. It is used to alleviate the knowledge acquisition problem by transforming distributed or continuous representations into discrete representations. From a practical perspective, it is used in situations where knowledge required to accomplish the task is not available and one intelligent technique depends upon another intelligent technique for its reasoning or processing. $x \Vdash y$ denotes that knowledge representation of intelligent technique x is transformed into knowledge representation of intelligent technique y.

In *FUSION* (Θ) relation, representation and information processing features of intelligent technique x are fused into the representation structure of another intelligent technique y. In this way the intelligent technique y augments its information processing in a manner which can cope with different levels of intelligence and information processing. From an application or practical viewpoint, this augmentation can be seen as a way by which an intelligent technique addresses its weaknesses and exploits its existing strengths to solve a particular real world problem. $x \Theta y$ denotes that intelligent technique x and intelligent technique y use the same knowledge representation but the centre around intelligent technique is x.

The practical hybrid technique models among ES, NN, FL, GA, and CBR are summarised in Table 3.2.

Table 3.2 Practical hybrid technique models

Intelligent Techniques	Practical Hybrid Models
ES	ES⋐NN; ES⋐GA; ES $\Vdash$ NN; ES Θ NN
NN	NN⋐FL⋐CBR; NN$\Vdash$ES; NN $\Vdash$FL; NNΘGA; NNΘES; NNΘFL
FL	FL⋐NN⋐GA; FL⋐ES; FL⋐NN; FL $\Vdash$ NN; FL Θ NN
GA	GA⋐NN; GA $\Vdash$ FL; GA $\Vdash$ NN; GAΘ ES; GAΘ FL; GAΘ NN
CBR	CBR⋐NN; CBR⋐GA; CBR $\Vdash$ NN; CBR Θ NN

3.2.2 Hybrid Strategy Models

Medsker (Medsker 1995) has identified five kinds of different hybrid strategy models: stand-alone model, transformational model, loose-coupling model, tight-coupling model, and fully-integration model as shown in Figure 3.2 (Medsker 1995).

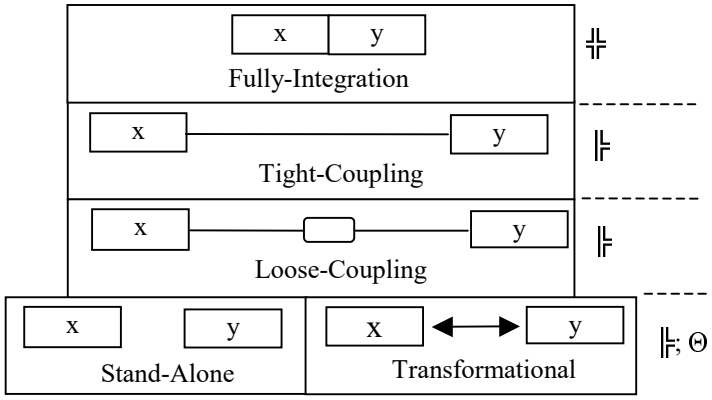

Figure 3.2 Models for integrating HIS

- **Stand-alone Models**

Stand-alone models consist of independent software components. These components do not interact in any way. While a degenerate case for integration purpose, the stand-alone model is an alternative worth discussing. The stand-alone approach uses independent systems to study different or the same aspects of an application problem with no prior commitment for the implementation technique of the final operational system. Exploratory work with each system can lead to a better understanding of an application and verification of knowledge to be incorporated into an intelligent system.

Developing stand-alone intelligent systems can have several advantages. Firstly, they provide direct means of comparing the problem-solving capabilities of the two techniques for a specific application. Secondly, used in parallel, the techniques provide redundancy in processing. Thirdly, developing one technique after finishing

a model of the other facilitates validating the prior development process. Finally, running two models in parallel permits a loose approximation of integration.

The structure of a stand-alone system consists of three levels: subsystems, results identifier, and results mediator as shown in Figure 3.3. A task is synchronously completed by all subsystems which are with different stand-alone intelligent techniques. The outputs of the subsystems are identified by the results identifier. The results mediator makes decision by analysing these identified results. Each subsystem can be implemented by an intelligent agent which can be dynamically added into or removed from the system. The results identifier and results mediator can be implemented by another two agents. The autonomy, reactivity and mental notions are important features of each intelligent agent. The features of interaction with the environment, hierarchical structure, intelligent agent as a member of the subsystem level, dynamic agents in the subsystem level, etc. associate stand-alone structure.

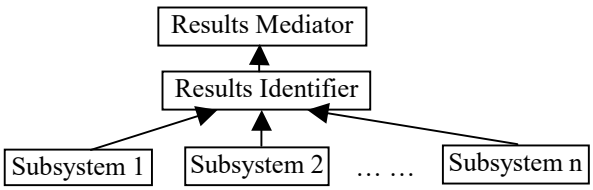

Figure 3.3 Structure of stand-alone systems

The benefits of a stand-alone model include the simplicity and ease of development using commercially-available software packages. On the other hand, stand-alone model development efforts of one technique are not transferable to the other, neither can support the weaknesses of the other technique, and the maintenance requirements are doubled. Both must be updated simultaneously to avoid confusion, and updates to one cannot help the other. Note that x and y in Stand-Alone model shown in Figure 3.2 can be any member of set X.

- **Transformational Models**

Transformational models are similar to stand-alone models in that the end result of development is an independent model that does not interact with the other. What distinguishes the two types of models is that transformational systems begin as one

type of system, and end up as the other. The transformational model entails a specific development sequence using, for example, an independent unsupervised neural network to discern from data the rules needed for a subsequence system that is to be implemented with fuzzy logic. Another example, similar to expert network development, is the translation of a fuzzy system to a neural network so that parameters for the final fuzzy system can be optimised *via* neural network training. The $\models$ and Θ relations adopted by the hybrid technique models are related to the transformational hybrid strategy models. In transformational model shown in Figure 3.2, (x, y) is a member of the set T (derived from Table 3.2):

$$T = \begin{Bmatrix} (ES, NN), (NN, ES), (NN, FL), (NN, GA), (FL, NN), \\ (GA, ES), (GA, FL), (GA, NN), (CBR, NN) \end{Bmatrix}$$

where: $T \subset X \times X$.

The structure of a transformational system consists of two levels: subsystems and results mediator as shown in Figure 3.4. Compared with the stand-alone systems, the results identifier is not useful in the transformational system because the subsystems themselves have transformed their results into the outputs with a uniform fashion. Each subsystem can also be implemented by an intelligent agent as described in stand-alone system. The results mediator can be implemented by another agent. The autonomy, reactivity and mental notions are important features of each intelligent agent. The features of interaction with the environment, multiple control, multiple interests, hierarchical structure, intelligent agent as a member of the subsystem level, dynamic intelligent agents in the subsystem level, etc. associate transformational structure.

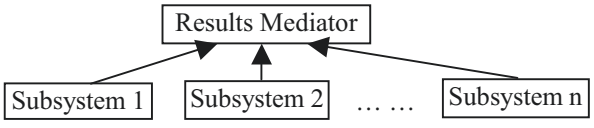

Figure 3.4 Structure of transformational systems

Transformational models offer several benefits to developers. They are often quick to develop and ultimately require maintenance on only one system.

Development occurs in the most appropriate environment. Similarly, the delivery technique offers operational benefits suitable to its environment.

Limitations to transformational models are significant. First, a fully automated means of transforming an intelligent technique *x* to an intelligent technique *y* is still needed. Also, significant modifications to the system may require a new development effort, which leads to another transformation. In addition to maintenance issues, the finished transformational system is limited operationally to the capabilities of the target technique.

- **Loose-Coupling Models**

Loose-coupling models are the first true form of integrating intelligent systems. The application is decomposed into separate intelligent system components that communicate *via* data files. Among the variations of loose-coupling models are pre-processors, post-processors, co-processors, and user interfaces. The $\Vdash$ relation adopted by the hybrid technique models is related to loose-coupling hybrid strategy models. In loose-coupling model shown in Figure 3.2, (x, y) is a member of the set *LC* (derived from Table 3.2):

$$LC = \begin{Bmatrix} (ES, NN), (NN, ES), (NN, FL), (FL, NN), \\ (GA, FL), (GA, NN), (CBR, NN) \end{Bmatrix}$$

where: $LC \subset X \times X$.

Compared to the more integrated intelligent system applications, loosely-coupled models are easy to develop. They are amenable to the use of commercially available intelligent system software, which reduces the programming burden on the developers. Both the system design and implementation processes are simplified with loose-coupling models. Finally, maintenance time is reduced because of the simplicity of the data file interface mechanism.

The structure of a loose-coupling system consists of three levels: components, manager, and application as shown in Figure 3.5. The manager level consists of task delegation and data management. The application level includes data files. Each application in the application level has a task which is completed by the components. This task can be decomposed into several subtasks which are delegated to specific components. These components cooperate to complete the task.

Each component which consists of an intelligent technique or a hybrid technique interacts with other ones by means of data files. The data files are manipulated by a data management. Each component can be implemented by an intelligent agent which is dynamically added into or removed from the system. The task delegation and data management are implemented by two or more agents. The autonomy, reactivity, pro-activeness, and mental notions are important features of each intelligent agent. The features of social ability, interaction with the environment, multiple control, multiple interests, subsystem, hierarchical structure, intelligent agent as a member of the component level, dynamic agents in the component level, application-based reorganisation, shared data management, etc. associate loose-coupling structure.

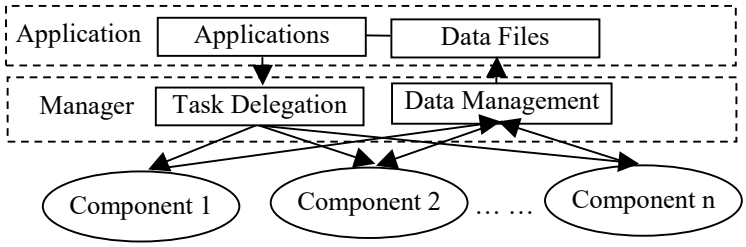

Figure 3.5 Structure of loose-coupling systems

Some limitations are associated with loose-coupling models. Because of the file-transfer interface, communications costs are high and operating time is longer. The development of the separate intelligent system components leads to redundancy of effort. Both must be capable of solving sub-problems in order to perform their unique computations, but because they lack direct access to each other's internal processing they must develop independent capabilities. This may also lead to overlap in the data input requirements and internal processing.

- **Tight-Coupling Models**

The categories of loose- and tight-coupling have significant overlap. Both can utilise the same set of independent intelligent techniques *x*, *y*, and etc. However, tight-coupling systems pass information *via* memory resident data structures rather than external data files. This improves the interactive capabilities of tight-coupling

models in addition to enhancing their performance. The $\models$ relation adopted by the hybrid technique models are related to tight-coupling hybrid strategy models. In tight-coupling model shown in Figure 3.2, (x, y) is a member of the set *TC* (derived from Table 3.2):

$$TC = \begin{Bmatrix} (ES, NN), (NN, ES), (NN, FL), (FL, NN), \\ (GA, FL), (GA, NN), (CBR, NN) \end{Bmatrix}$$

where: $TC \subset X \times X$.

Tight-coupling models can function under the same variations as loose-coupling models, except that the tight-coupling versions of pre-, post- and co-processors are typically faster. The structure of a tight-coupling system consists of three levels: components, manager, and application as shown in Figure 3.6. The manager level consists of task delegation and message management. The application level includes messages. Compared with the loose-coupling systems, the difference is that each component in tight-coupling system interacts with other ones by means of messages rather than data files. The messages are organised by the message management. Each component can be implemented by an intelligent agent as described in loose-coupling system. The message manager can be involved in each intelligent agent. The messages can be carried by agent communication language, such as, KQML (Finin, Labrou et al. 1997). The autonomy, reactivity, pro-activeness, and mental notions are important features of each intelligent agent. The features of social ability, interaction with the environment, multiple control, hierarchical structure, intelligent agent as a member of the component level, dynamic agents in the component level, application-based reorganisation, etc. associate tight-coupling structure.

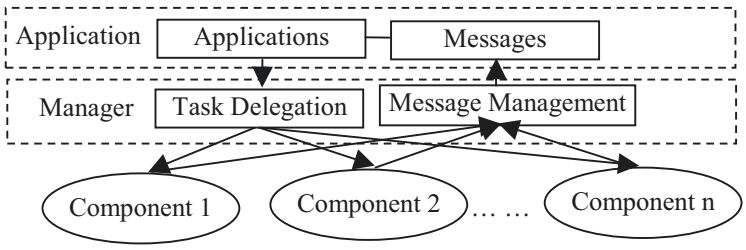

Figure 3.6 Structure of tight-coupling systems

The tight-coupling model has the benefits of reduced communication overhead and improved runtime performance compared to loose-coupling. Several commercial packages are suitable for developing tight-coupling models and maintaining the modularity of expert system and neural network components. Overall, tight-coupling model offers design flexibility and robust integration.

Tight-coupling systems have three principle limitations. First, the development and maintenance complexity increases due to the internal data interface. Second, tight-coupling model suffers from redundant data gathering and processing, just like loose-coupling. Once again, this is due to the independence of the intelligent system components. Finally, the verification and validation process is more difficult, particularly for embedded applications.

- **Fully-Integration Models**

Fully-integration systems share data structures and knowledge representations. Communication between the different components is accomplished *via* the dual nature of the structures. Reasoning is accomplished either cooperatively or through a component designated as the controller. Several variations of fully-integration systems exist, including connectionist systems, the utilisation of I/O nodes, sub-symbolic to symbolic connectivity, and integrated control mechanisms. Fully-integration systems are unified systems in which fuzzy techniques are embedded in the neural network, for example in the use of fuzzy neurons. Many research and development projects are exploring various ways to fuzzify the internal operations of neural networks. The $\fatslash$ relation adopted by the hybrid technique models is related to fully-integration hybrid strategy models. In fully-integration model shown in Figure 3.2, (x, y) is a member of the set F (derived from Table 3.2):

$$F = \left\{ \begin{array}{l} (ES, NN), (ES, GA), (ES, FL), (NN, ES), (NN, FL), (NN, CBR), \\ (NN, GA), (FL, NN), (FL, CBR), (FL, ES), (GA, ES), (GA, NN), \\ (GA, CBR), (CBR, NN), (CBR, FL), (CBR, GA) \end{array} \right\}$$

where: $F \subset X \times X$.

The structure of a fully-integration system consists of three levels: components, manager, and application as shown in Figure 3.7. The manager level consists of knowledge base manipulator and task delegation. The application level includes

knowledge base. Compared with the loose-coupling and tight-coupling systems, the difference is that all components in fully-integration system share a uniform knowledge base instead of each component with its own knowledge base. Each component can be implemented by an intelligent agent as described in loose-coupling system. The task delegation and knowledge base manipulator are implemented by two or more agents. The autonomy, reactivity, and mental notions are important features of each intelligent agent. The features of social ability (commitments), interfaces with other entities, interaction with the environment, hierarchical structure, intelligent agent as a member of the component level, dynamic agents in the component level, application-based reorganisation, shared data management, etc. associate fully-integration structure.

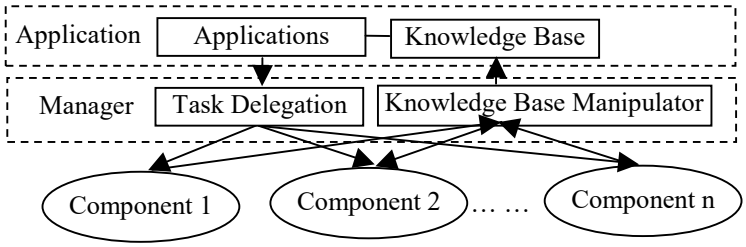

Figure 3.7 Structure of fully-integration systems

The benefits of fully-integration include robustness, improved performance, and increased problem solving capabilities. Robustness and performance improvements stem from the dual nature of the knowledge representations and data structures. In addition, little or no redundancy occurs in the development process. Finally, fully-integration models can provide a full range of capabilities not found in non-integrated models.

Fully-integration model has limitations caused by the increased complexity of the inter-module interactions. Specifying, designing, and building fully-integration models is complex, tools that facilitate fully-integration models are distinctly lacking on the market, and verifying, validating, and maintaining fully-integration systems are issues for further research and development.

3.2.3 Characteristics of HIS

HIS is complicated because HIS represents not only a combination of different intelligent techniques, but also the integration of intelligent techniques with legacy computing systems or programs (Zhang and Zhang 2004). However, the complexity of HIS exhibits a number of important regularities (Simon 1996).

1) HIS consists of a number of inter-related subsystems or components organised in a hierarchical fashion. At any given level, subsystems or components work together to achieve the functionality of their parent system. Moreover, within a subsystem, the constituent components work together to deliver overall functionality. Thus, the same basic model of interesting components, working together to achieve particular objectives, occurs throughout the system.

2) The choice of the primitive components in each hierarchical level is arbitrary and is defined according to the needs of observers. At one level, entire subsystems can be viewed as a singleton; a collection of processes can be viewed as a primitive component; and so on; until the system eventually bottoms out.

3) The primitive components in each hierarchical level may be dynamic at unpredictable time. Some structures are needed to provide a variety of stable intermediate forms, which are essential for rapid development of complex HIS. Their availability means that individual components or organisational groupings can be developed in relative isolation, and then added into the system in an incremental manner. This, in turn, ensures there is a smooth growth in functionality.

4) The interactions between primitive components may occur at unpredictable times and for unpredictable reasons. It is possible to distinguish between the interactions among subsystems and the interactions within subsystems. The latter are both more frequent and more predictable than the former. This gives rise to the view that HIS is nearly decomposable, that is, subsystems can be treated almost as if they are independent of one another, but not quite, since they are some interactions between them. Moreover, although many of these interactions can be predicted at design time, some cannot. So some interaction mechanisms are needed

to manage the interactions between primitive components at run-time rather than design time.

3.3 Methodologies and Hybrid Strategies

Although many agent-oriented methodologies have been developed so far, they are deficient in HIS construction because HIS possesses their own distinct characteristics rather than general agent-based systems. There is no one methodology that can fully meet the requirements of the analysis and design of agent-based HIS (Zhang and Zhang 2004). In this section, six well-known agent-oriented methodologies are evaluated with the characteristics of HIS. The deficiencies of those methodologies for constructing HIS are clarified. At the same time, MAS-CommonKADS is selected to be tailored for proposing MAHIS, after ranking the six well-known agent-oriented methodologies according to the attributes of each hybrid strategy.

3.3.1 Comparison of Agent-Oriented Methodologies

We start an evaluation to clarify the merits and limitations of the six well-known methodologies in the analysis and design of agent-based HIS. Because agent-based HIS is of the specific characteristics of HIS as well as the characteristics of the general MAS, generic evaluation methods with necessary enrichment can be used to evaluate the existing agent-oriented methodologies. The idea of *attributes tree* (Cernuzzi and Rossi 2002) is borrowed to abstract the characteristics of HIS and to compare and rank these agent-oriented methodologies according to the characteristics of each hybrid strategy. The reason to select the *attributes tree* evaluation framework is that it is known as a qualitative analysis followed by a quantitative rating and it is very convenient to extend the *attributes tree*. To extend the *attributes tree* just adds some defined roots or braches. Furthermore, this

framework is cited by some valuable evaluation frameworks (Dam and Winikoff 2003, Sturm and Shehory 2003, Sudeikat, Braubach et al. 2004).

The evaluation framework proposed by Cernuzzi and Rossi is enriched with *reorganisation attributes* according to the characteristics of HIS. In the extended evaluation framework, the attributes are grouped into four different categories: those concerning the own characteristics of agents (*internal attributes*), those referred to the interaction process (*interaction attributes*), those standing for the reorganisation of agents (*reorganisation attributes*), and those more directly inherent to the design and development process (*other process requirements*). The attributes tree model is shown in Table 3.3.

Table 3.3 Enriched attributes tree model

A. Internal attributes	*B.* Interaction attributes	*C.* Reorganisation attributes	*D.* Other process requirements
A.1 Autonomy	*B.1* Social ability	*C.1* Hierarchical structure	*D.1* Modularity
A.2 Reactivity	*B.1.1* Organisational relationships among agents	*C.1.1* Agent categories	*D.1.1* Decomposition
A.3 Pro-activeness	*B.1.2* Interaction with others agents	*C.1.2* Organisational structure	*D.1.2* Models' dependence
A.4 Mental notions	*B.1.2.1* Types of agents interaction	*C.1.3* Coordination mechanism	*D.2* Abstraction
A.4.1 Beliefs	*B.1.2.2* commitments	*C.2* Agent as a level member	*D.2.1* Abstraction inside each phase
A.4.2 Goals (Desires)	*B.1.3* Conversations with other agents	*C.3* Dynamic agents in each level	*D.2.2* Existence of design primitives and high level abstraction mechanism
A.4.3 Intentions	*B.1.4* Interfaces with other entities	*C.4* Application-based reorganisation	*D.3* System view
	B.2 Interaction with the environment	*C.5* shared items management	*D.4* Communication support
	B.3 Multiple control	*C.5.1* Data	*D.4.1* clear and precise models
	B.4 Multiple Interests	*C.5.2* Legacy systems	*D.4.2* Systematic transition
	B.5 Subsystem interaction	*C.5.3* Agents	

The *internal attributes* include: *autonomy*, *reactivity*, *pro-activeness*, and *mental notions*. The *interaction attributes* include *social ability*, *interaction with the environment*, *multiple control*, *multiple interests*, and *subsystems interaction*. The *reorganisation attributes* include *hierarchical structure*, *agent as a level member*, *dynamic agents in each level*, *application-based reorganisation*, and *shared items management*. The *"hierarchical structure"* indicates whether a methodology has the capability to elaborate the hierarchical relationships of agents in a developed system. The hierarchical relationships include agent categories, organisational structure, and coordination mechanism. The *"agent as a level member"* indicates whether a methodology has the capability to elaborate the primitive members in each level of the hierarchical structure. The *"dynamic agents in each level"* indicates whether a methodology has the capability to elaborate the dynamic addition and removal of the primitive members in each level of the hierarchical structure. The *"application-based reorganisation"* indicates whether a methodology has the capability to elaborate the dynamic applications which may be generated by reorganising the agents at run-time. The *"shared items management"* indicates whether a methodology has the capability to elaborate the shared resources, e.g., databases, legacy software, and agents. The *other process requirements* include *modularity*, *abstraction*, *system view*, and *communication support*. It seems evident that a good methodology may offer to agent-based system designers a set of models, techniques and mechanisms that possibly cover all the attributes in the most exhaustive way.

During evaluation of methodologies, each attribute is assigned with a score and the score of attributes on the node is calculated based on those of their children. The score assigned to each attribute is in the range of 0 to 10.

In order to carry out an independent analysis for a specific methodology as well as a comparative analysis, six methodologies, Gaia (G), MAS-CommonKADS (Mc), MaSE (Ms), ODAC (O), Prometheus (P), and Tropos (T), selected and introduced in section 2.2 have been evaluated with the enriched *attributes tree*. The comparative results are presented in Table 3.4. The number in Table 3.4 is the mapping value of each attribute against a methodology (see Chapter 8 for the rules to work out the numbers).

Table 3.4 The evaluation results with attributes tree

Attributes	G	Mc	Ms	O	P	T
A	**7.83**	**8.33**	**8.1**	**7.6**	**8.6**	**7.4**
A.1	10	10	10	10	10	10
A.2	10	10	10	10	10	10
A.3	3	5	5	5	5	3
A.4	8.33	8.33	7.7	6	10	6.7
A.4.1	5	5	3	3	10	5
A.4.2	10	10	10	5	10	10
A.4.3	10	10	10	10	10	5
B	**4.7**	**7**	**4.9**	**3.9**	**4.3**	**3.3**
B.1	8.3	10	6.3	6.3	6.3	5.6
B.1.1	8	10	5	5	5	5
B.1.2	10	10	10	10	10	7.5
B.1.2.1	10	10	10	10	10	10
B.1.2.2	10	10	10	10	10	5
B.1.3	10	10	10	10	10	10
B.1.4	5	10	0	0	0	0
B.2	5	5	3	3	5	3
B.3	5	5	3	0	0	3
B.4	5	5	5	5	5	5
B.5	0	10	10	5	5	0
C	**0**	**1.33**	**0**	**0**	**0**	**0**
D	**7.5**	**9.5**	**9.5**	**8.1**	**8.3**	**7.9**
D.1	10	8.2	8	10	8.2	8.9
D.1.1	10	10	10	10	10	10
D.1.2	10	6.4	6	10	6.4	7.8
D.2	10	10	10	5	7.5	10
D.2.1	10	10	10	5	5	10
D.2.2	10	10	10	5	10	10
D.3	0	10	10	10	10	5
D.4	10	10	10	7.5	7.5	7.5
D.4.1	10	10	10	10	10	5
D.4.2	10	10	10	5	5	10
Total	**20.0**	**26.2**	**22.5**	**19.6**	**21.2**	**18.6**

From the evaluation results, we know that all methodologies are deficient in *reorganisation attributes* which denote the capability of a methodology to construct agent-based HIS. Only MAS-CommonKADS has the capability of *organisational structure* (score 10) and *coordination mechanism* (score 10) designs, so the value of the *hierarchical structure* is 6.67 ((0+10+10)/3=6.67) and the value of the *reorganisation attributes* is 1.33 ((6.67+0+0+0+0)/5=1.33). However, all methodologies are strong in the *internal attributes* and *other process requirements*. MAS-CommonKADS and Prometheus are better in the *internal attributes*. MAS-CommonKADS is better than others in the *interaction attributes*. MAS-CommonKADS and MaSE are better than other methodologies in the *other process requirements*. However, from the comprehensive attributes viewpoint, MAS-CommonKADS is better than others.

3.3.2 Ranking Methodologies for Selection

For selecting a methodology for tailoring, the methodologies are ranked first according to the attributes associated different strategy. The relationships between the attributes and the hybrid strategies can be analysed according to the characteristics of each hybrid strategy. The five attributes, namely, *hierarchical structure*, *agent as a level member*, *dynamic agents in each level*, *application-based reorganisation*, and *shared items management*, evaluate the capabilities of a methodology in constructing agent-based HIS with dynamic organisation and hierarchical structure in accordance with the characteristics of HIS.

The *stand-alone* hybrid strategy associates the following attributes: *autonomy, reactivity, mental notions, interaction with the environment, hierarchical structure, agent as a level member, dynamic agents in each level, and other process requirements* according to the characteristics of the stand-alone systems. The transformational hybrid strategy associates the following attributes: *autonomy, reactivity, mental notions, interaction with the environment, multiple control, multiple interests, hierarchical structure, agent as a level member, dynamic agents*

in each level, and *other process requirements*. The loose-coupling hybrid strategy associates all attributes. The tight-coupling hybrid strategy associates the following attributes: *internal attributes, social ability, interaction with the environment, multiple control, reorganisation, and other process requirements*. The fully integration hybrid strategy associates the following attributes: *autonomy, reactivity, mental notions, social ability (commitments), interfaces with other entities, interaction with the environment, reorganisation, and other process requirements*.

The following formula is utilised to calculate the *ranking value* of each methodology against a specific hybrid strategy model.

$$R_{ij} = \frac{1}{N_i} \sum_{k=1}^{Ni} V_{jk \to x}$$

where: R_{ij} denotes the average of the attributes' values of the *hybrid strategy i* against the *methodology j*;

$V_{jk \to x}$ denotes the *evaluation value* of the k^{th} *attribute* associated the *hybrid strategy i*; The k^{th} *attribute* can be mapped to the *attribute x* of the *methodology j*;

N_i denotes the number of attributes associated the *hybrid strategy i*;

i = 1(*stand-alone model*), 2 (*transformational model*), 3 (*loose-coupling model*), 4 (*tight-coupling model*), 5 (*fully-integration model*);

j = 1 (Gaia), 2 (MAS-CommonKADS), 3 (MaSe), 4 (ODAC), 5 (Prometheus), 6 (Tropos);

N_1 = 8 because eight attributes associated the stand-alone model; $k \to x$ = 1→*A.1* (*autonomy*), 2→*A.2* (*reactivity*); 3→*A.4* (*mental notions*); 4→*B.2* (*interaction with the environment*); 5→*C.1* (*hierarchical structure*); 6→*C.2* (*agent as a level member*); 7→*C.3* (*dynamic agents in each level*); 8→*D* (*other process requirements*);

N_2 = 10 because ten attributes associated the transformational model; $k \to x$ = 1→*A.2* (*reactivity*), 2→*A.4* (*mental notions*); 3→*B.1* (*social ability*); 4→*B.2* (*interaction with the environment*); 5→*B.3* (*multiple control*); 6→*B.4* (*multiple interests*); 7→*C.1* (*hierarchical structure*);

8→*C.2* (*agent as a level member*); 9→*C.3* (*dynamic agents in each level*); 10→*D* (*other process requirements*);

N_3 = 4 because four attributes associated the loose-coupling model; $k \rightarrow x$ = 1→*A* (*internal attributes*), 2→*B* (*interaction attributes*); 3→*C* (*reorganisation attributes*); 4→*D* (*other process requirements*);

N_4 = 6 because six attributes associated the tight-coupling model; $k \rightarrow x$ = 1→*A* (*Internal attributes*), 2→*B.1* (*social ability*); 3→*B.2* (*interaction with the environment*); 4→*B.3* (*multiple control*); 5→*C* (*reorganisation attributes*); 6→*D* (*other process requirements*);

N_5 = 8 because eight attributes associated the fully-integration model; $k \rightarrow x$ = 1→*A.1* (*autonomy*), 2→*A.2* (*reactivity*); 3→*A.4* (*mental notions*); 4→*B.1.2.2* (*commitments*); 5→*B.1.4* (*interfaces with other entities*); 6→*B.2* (*interaction with the environment*); 7→*C* (*reorganisation attributes*); 8→*D* (*other process requirements*);

The ranking value of each methodology against a hybrid strategy model can be calculated according to the evaluation values of the relevant attributes in Table 3.4 by using this formula. For example, the ranking value of Gaia against the stand-alone model can be calculated as:

$$R_{11} = (10+10+8.33+5+0+0+0+7.5)/8 = 5.10$$

The following are the results after calculating:

R_{11}: ***5.10***; R_{12}: ***6.19***; R_{13}: ***5.02***; R_{14}: ***4.64***; R_{15}: ***5.41***; R_{16}: ***4.70***

R_{21}: ***4.91***; R_{22}: ***5.96***; R_{23}: ***4.45***; R_{24}: ***3.84***; R_{25}: ***4.46***; R_{26}: ***4.12***

R_{31}: ***5.00***; R_{32}: ***6.21***; R_{33}: ***5.63***; R_{34}: ***4.90***; R_{35}: ***5.30***; R_{36}: ***4.65***

R_{41}: ***5.61***; R_{42}: ***6.31***; R_{43}: ***4.98***; R_{44}: ***4.17***; R_{45}: ***4.70***; R_{46}: ***4.48***

R_{51}: ***6.98***; R_{52}: ***7.85***; R_{53}: ***6.28***; R_{54}: ***5.89***; R_{55}: ***6.66***; R_{56}: ***5.33***

The methodologies for each hybrid strategy model can be ranked according the *ranking values* as shown in Table 3.5. The main limitations for each hybrid strategy model are the weak attributes of the first methodology in the rank. For example, the limitations of MAS-CommonKADS for the loose-coupling systems are *reorganisation attributes (C)*, pro-*activeness (A.4.1)*, *interaction with the environment (B.2)*, *multiple control (B.3)*, *multiple interests (B.4)*, and *Models'*

dependence (D.1.2). From Table 3.5 we know that MAS-CommonKADS is better than other methodologies for constructing all kinds of HIS. So MAS-CommonKADS can be selected as the methodology for tailoring in order to propose MAHIS.

Table 3.5 Ranked methodologies for hybrid strategies

Strategy models	Ranking	Ranked main limitations
Stand-Alone	**Mc**, P, G, Ms, T, O	*C.2; C.3; B.2; B.3; B.4; D.1.2*
Transformational	**Mc**, G, P, Ms, T,O	*C.2; C.3; B.2; B.3; B.4; D.1.2*
Loose-Coupling	**Mc**, Ms, P, G, O, T	*C; A.3; A.4.1; B.2, B.3; B.4; D.1.2*
Tight-Coupling	**Mc**, G, Ms, P, T, O	*C; A.3; A.4.1; B.2, B.3; D.1.2*
Fully-Integration	**Mc**, G, P, Ms, O, T	*C; A.4.1; B.2; D.1.2*

3.4 Summary

Before discussing methodologies for agent-based HIS construction, the hybrid techniques, hybrid integration strategies (hybrid strategies), and hybrid modelling must be made clear first. The hybrid technique models and hybrid strategy models can help to understand and analyse the characteristics of HIS. To avoid building MAHIS from scratch, to tailor a complete and well-ground agent-oriented methodology is wise. The gap between the existing well-known agent-oriented methodologies and agent-based HIS construction can be identified by evaluating these methodologies. All discussions in this chapter are the foundation of the MAHIS development in next chapter. Further discussion is needed to detail for proposing MAHIS by:

- Bridging the gap between MAS-CommonKADS and agent-based HIS construction;
- Cutting out the redundant contents of MAS-CommonKADS which are not suitable for constructing agent-based HIS or conflict with the extended parts;
- Proposing MAHIS based on the tailored MAS-CommonKADS.

Chapter 4

4 MAHIS: A Methodology for Constructing Agent-Based HIS

In Chapter 3 it is clarified that even the best methodology MAS-CommonKADS cannot cover all the characteristics of HIS. The distinct deficiency is that MAS-CommonKADS almost has no reorganisation ability which is considered to be the foundation of the agent-based HIS. However, there is mismatching between the detailed requirements of a methodology and the dynamics of HIS. The methodology usually needs enough explicit user requirements and constraint conditions for comprehensive analysis and design of HIS. On the other hand, the dynamic addition and removal of some components are essential abilities of HIS according to the characteristics of HIS. How to develop a methodology for supporting agent-based HIS construction based on the definite user requirements and constraint conditions is a problem.

We try to solve this problem from a platform perspective. According to the definite user requirements and constraint conditions, a platform is developed first. All applications or agents can be dynamically added into or removed from the platform by following the architectural model of the platform. So, the proposed methodology must take the platform construction into account.

Following this idea, we have proposed the methodology MAHIS. To avoid building MAHIS from scratch, we have extended MAS-CommonKADS with the capabilities in agent-based HIS constriction. MAHIS includes a *process life cycle*, a set of modelling languages, and a set of techniques as shown in Figure 1.2. So far MAHIS consists of three process stages: conceptualisation, analysis, and design. Each process stage includes one or more models.

Section 4.1 presents the *process life cycle*: hybrid system development life cycle (HSDLC). The framework of MAHIS is introduced in Section 4.2. The eight models of MAHIS are discussed in the following three sections in accordance with the three process stages: conceptualisation (Section 4.3), analysis (Section 4.4), and design (Section 4.5).

4.1 Hybrid System Development Cycle

HSDLC should provide a set of activities to construct and manage HIS in several lifecycle stages. The lifecycle stages are usually defined as requirements' gathering, analysis, design, implementation, and testing (Sturm and Shehory 2003). Requirements' gathering is the stage of the lifecycle in which the specification (usually in free text) of the necessities from the system is done. Analysis is the stage of the lifecycle that describes the observable characteristics of the system, e.g., functionality, performance, and capacity. Design is the stage of the lifecycle that defines the way in which the system will accomplish its requirements. The models defined in the analysis stage are either refined, or transformed, into design models that depict the logical and the physical nature of the software product. Implementation is the stage of the lifecycle that converts the developed design models into software executable within the system environment. This either involves the hand coding of program units, the automated generation of such code, or the assembly of already built and tested reusable code components from an in-house reusability library. Testing focuses on ensuring that each deliverable from each stage conforms to, and addresses, the stated user requirements.

Besides the usual definition of the lifecycle stages, the following factors should be taken into account while proposing the HSDLC. Firstly, the analysis, design, and implementation of platform and hybrid intelligent applications are separate. However, the development of the platform is related to the hybrid intelligent applications. On the other hand, the hybrid intelligent applications are based on the platform. Because agent-based HIS is dynamic, the platform should support dynamic addition and removal of agents. The platform needs to organise all agents in agent categories with hierarchical structure. The member of each agent category may be a set of agents (agent group). The agent groups can be dynamically added into or removed from the platform at run-time. Secondly, because MAHIS is proposed based on MAS-CommonKADS, the distinct features of the MAS-CommonKADS lifecycle must be considered. The lifecycle stages of MAS-CommonKADS include conceptualisation, analysis and design. Finally, the characteristics of HIS should be included in the HSDLC. There are three hybrid technique relations and five hybrid strategy models as discussed in Section 3.2. When the real-world complex problem requirements are obtained, developers can determine: what intelligent techniques should be taken; which hybrid technique relations should be used for hybridising those selected intelligent techniques; what hybrid strategy should be adopted. By synthetising all aforementioned factors, the HSDLC is proposed as in Figure 4.1.

The lifecycle stages of the HSDLC include conceptualisation, analysis, design, implementation, and maintenance. The conceptualisation stage includes two steps: real-world complex problem requirements and hybrid technique and hybrid strategy analysis. The task of the conceptualisation is to obtain a description of the problem and to determine use cases which can help to understand informal requirements and to test the system. The analysis stage includes three steps: agent categorisation with hierarchical structure, definition of the members of each category, and platform analysis and agent application analysis. The design stage includes platform design and application design. The implementation stage includes platform implementation, agent coding and testing, and integrating applications to the platform. Maintenance stage includes operation and maintenance.

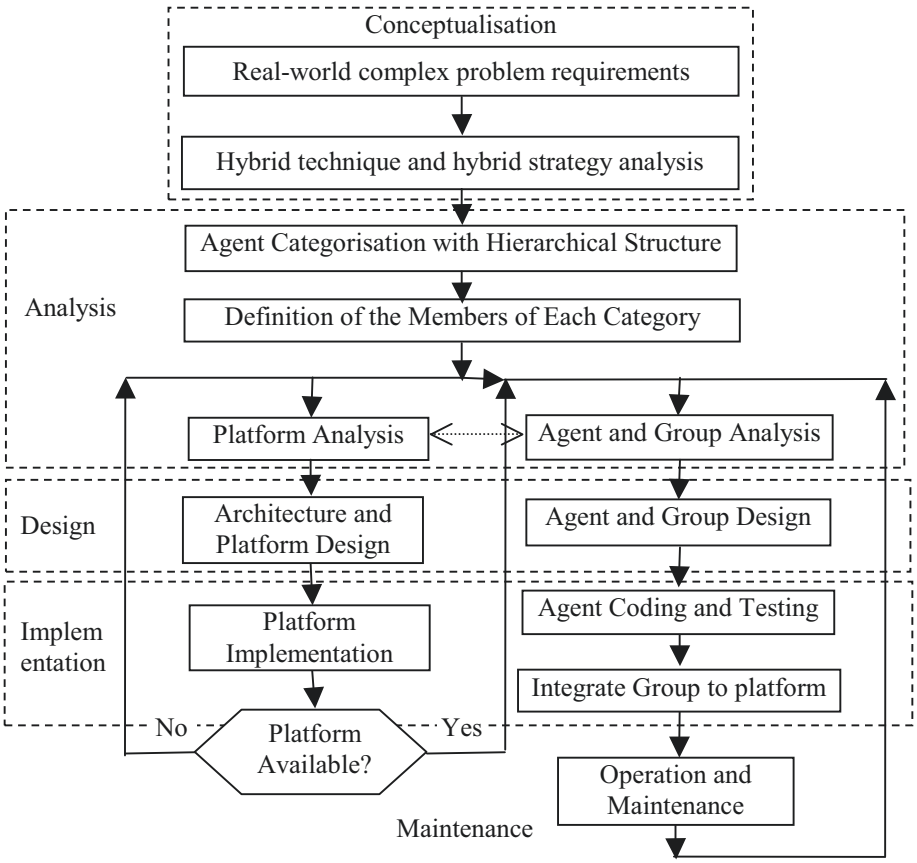

Figure 4.1 Hybrid system development life cycle

- Real-world complex problem requirements. The original documents (the initial tables, data, processing, or other information) to describe the real-world complex problem.
- Hybrid technique and hybrid strategy analysis. To determine intelligent techniques, hybrid technique relations, and hybrid strategies.
- Agent categorisation with hierarchical structure. To categorise agents and make the categories in a hierarchical structure.
- Definition of the members of each category. To group agents in a category and make the groups as the members of the category.
- Platform analysis and design. To develop a platform organised all agents dynamically. The dynamic primitive component is agent group.

- Agent and group analysis. Determination of the requirements of the system starting from the problem statement. This stage consists of delimitation, decomposition, and validation (Iglesias, Garijo et al. 1996).
- Agent and group design. How the requirements of the analysis phase can be achieved by the developing of the design model is determined here. The architecture of each group and each agent is also determined. This phase consists of group-based application design and architecture design.
- Agent coding and testing. This is an open question, depending on the usage of a general purpose language or a formal agent language, which can be directly executable or translated to an executable form.
- Integrate agent group to platform. Registration or cancellation of an agent group. There is at least one agent in each agent group to be in charge of the business of registration and cancellation.
- Operation and maintenance. The correct behaviour of the global system can be partially tested by using typical scenarios which deal with the possible conflicts and the methods for conflict resolution. Since the global behaviour of the system cannot be determined during analysis or design, simulation of the behaviours is usually needed (Iglesias, Garijo et al. 1996).

4.2 Framework of MAHIS

A methodology should include three aspects of components: lifecycle process, technique set and modelling language (Sturm and Shehory 2003). Lifecycle coverage of a particular methodology involves ascertaining what elements of software development are dealt with within the methodology. Each methodology may have elements that are useful to several stages of the development lifecycle (Sturm and Shehory 2003). Having the development stages defined is not sufficient for using a methodology. A methodology should further elaborate the activities within the development lifecycle, in order to provide the user of the methodology with the means of using it properly and efficiently. Providing a detailed description

of the various activities during the development lifecycle would enhance the appropriate use of a methodology and increase its acceptability as a well-formed engineering approach.

The technique set includes the techniques or methods which complete the tasks of a proposed system. They include the techniques that have been tried and tested over the last decade; but also may include new techniques that are more experimental (Graham, Henderson-Sellers et al. 1997). Some indication on the level of maturity of the individual technique is thus given as part of its full specification. The modelling language consists of meta-model and notation adopted by a methodology. A methodology needs to include a means for representing the generated artefacts; it needs to contain a notational element.

Under considering the components of a methodology, we have proposed MAHIS based on MAS-CommonKADS. The models and their relationships are presented in Figure 1.1 (see Section 1.3).

MAHIS consists of eight models: *Hybrid Strategy Identification Model*, *Organisation Model*, *Task Model*, *Agent Model*, *Expertise Model*, *Coordination Model*, *Reorganisation Model*, and *Design Model*. The MAHIS models are grouped into three levels: conceptualisation level, analysis level, and design level in accordance with the first three process stages of the HSDLC. That is, MAHIS includes three process stages at the moment. The *conceptualisation level* includes the *Hybrid Strategy Identification Model* and the description of hybrid problem requirements. The *Hybrid Strategy Identification Model* corresponds to the hybrid technique and hybrid strategy analysis in the HSDLC. During this phase, an elicitation task to obtain a preliminary description of the hybrid problem is carried out. Based on the problem description, the hybrid strategy model adopted by the HIS is identified. The purpose to identify the hybrid strategy is to help other models to decide the architectural model of HIS because the hybrid strategy adopted by a hybrid intelligent system decides the organisational structure and coordination mechanism of the system.

In fact, HIS with different hybrid strategies decides the distribution of agents and the architectural model which organises agents in the agent-based HIS. The

distribution of agents can be graded into three levels: LOW, MIDDLE, and HIGH. The LOW means that the agents can be only distributed in a small range by a local network, e.g., in a laboratory or a department. The MIDDLE means that the agents can be distributed in a larger range by interconnected local networks, e.g., in an institution or a company. The HIGH means that agents can be distributed in a large range by Internet. The architectural model consists of organisational structure and coordination mechanism. The organisational structure presents the interrelationship among agents in a system. The organisational structures can be modelled into four types: Peer-to-Peer, Tree, Grouping with facilitator (Grouping), and Ring. The coordination mechanism is the protocols to manage inter-dependencies between the activities of agents. The coordination mechanisms can be classified into five patterns: Direct Search, Matchmaker, Broker, Contract-net, and Token Ring. The architectural models can be proposed by combination of the organisational structures and coordination mechanisms. Suitable architectural models for each hybrid strategy are presented in Table 4.1.

Table 4.1 Suitable architectural models for hybrid strategies

Hybrid Strategies	Organisational Structure	Coordination Mechanism
Stand-alone	Peer-to-Peer	Direct-search; Contract-net
Transformational	Peer-to-Peer	Direct-search; Contract-net
Loose-coupling	Grouping; Ring	Matchmaker; Token-ring
Tight-coupling	Tree; Ring	Matchmaker; Token-ring
Fully-integration	Tree	Broker

The analysis level of MAHIS includes the *Organisation Model*, *Task Model*, *Agent Model*, *Reorganisation Model*, *Expertise Model*, and *Coordination Model*. These models can be divided into two sublevels: context and concept. The context sublevel includes the *Organisation Model*, *Task Model*, and *Agent Model*, which attempt to clarify the tasks, agents, organisational context, and environment. The concept sublevel includes the *Reorganisation Model*, *Expertise Model*, and *Coordination Model*, which issue the conceptual descriptions of the knowledge applied in a task, the interactions between agents, and the hierarchical structure and

the primitive members. The design level only includes the design model which consists of four steps: architecture design, agent communication language (ACL) design, platform design, and application design.

The hybrid problem to be solved is represented in the *Hybrid Problem Requirements*. The information in the *Hybrid Problem Requirements* can be used to develop the *Hybrid Strategy Identification Model*, *Organisation Model*, *Task Model*, and *Agent Model*. The *Organisation Model* supports the analysis of the major features of an organisation in order to describe the organisation into which the HIS is going to be introduced and the social organisation of the agent society. The *Task Model* analyses the global task layout, its inputs and outputs, preconditions and performance criteria, as well as needed resources and competences. The *Agent Model* describes the agent characteristics: groups and hierarchy. The purpose of the *Expertise Model* is to explicate in detail the types and structures of the knowledge used in performing a task. The *Coordination Model* describes the conversations between agents: their interactions, protocols, and required capabilities. The *Reorganisation Model* describes the management mechanism of hierarchical structure of the system and the primitive members in each level. The dynamic addition, removal, and reuse of agents are described in this model. The *Design Model* gives the technical system specification in terms of application, architecture, platform, and agent communication language to concretise the outputs of the reorganisation, coordination, and expertise models. The output of the design model can be implemented based on the different developing environments.

4.3 Conceptualisation

This is the first process stage of MAHIS. The task of the conceptualisation is to obtain a first description of the problem and to determine use cases which can help to understand informal requirements and to test the system. The method of use case modelling has the advantage of being formalised with Message Sequence Charts,

which are used for modelling the proposed coordination model. This stage includes hybrid problem requirements and *hybrid strategy identification model*.

4.3.1 Hybrid Problem Requirements

During this phase, an elicitation task to obtain a preliminary description of the problem is carried out following a user-centred approach by determining some use cases (scenarios) which can help us to understand informal requirements and to test the system. Use cases are described using OOSE notation (Iglesias, Garijo et al. 1997) and the interactions are formalised with MSC (Message Sequence Charts) (Rudolph, Graubman et al. 1996).

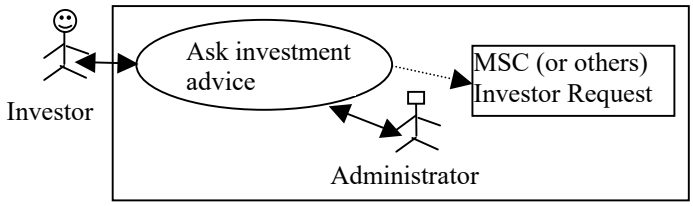

Figure 4.2 Use case notation

Let us have a look of one example for easily understanding MAHIS. We can identify one user role: *the investor*, a person who wishes to invest in some fields, e.g., stock market, real estate, bank deposit. When an investor wants to invest some money, he/she usually goes to a financial investment adviser for advice. The first thing the adviser needs to do is to understand the investor's individual circumstance (*IIC*). Based on the information of investor's individual circumstance, the adviser will evaluate the financial risk tolerance ability as well as the investor's investing goals (*IIG*). The *IIG* may be one of income, growth, and avoid risk (Zhang and Zhang 2004). If the *IIG* is income, investments that provide interest or dividend payments regularly and dependably are required. Two scenarios are identified: the system answers with an available investing field (*investing-field*) or with no available investing field and the cause. If there is no available investing field, the investor can change the information of *IIC* or/and *IIG*. The interaction between the

user and the system is represented in Figure 4.2 by using the use case notation of OOSE (Iglesias, Garijo et al. 1997).

The interactions of the use cases are formalised using MSC (Rudolph, Graubman et al. 1996), packages of AUML, templates of AUML (Odell, Parunak et al. 2000, Odell, Parunak et al. 2001), or protocol diagrams (Bauer, Muller et al. 2001) as a notation. Figure 4.3 presents the investor request in MSC notation. In this figure two message interchange alternatives are combined with the alternative (alt) operator. A basic MSC contains the description of the asynchronous communication between entities called instances, and has primitives for local actions, timers (set, reset and time-out), process creation, process stop, co-regions, and inline operators expressions for composition of event structures (alternative, parallel composition, iteration, exception and optional regions). The purpose in this phase is to get an idea of the interactions, but they will be refined later in the coordination model, specifying the data/knowledge interchanged and the speech-act of each interaction.

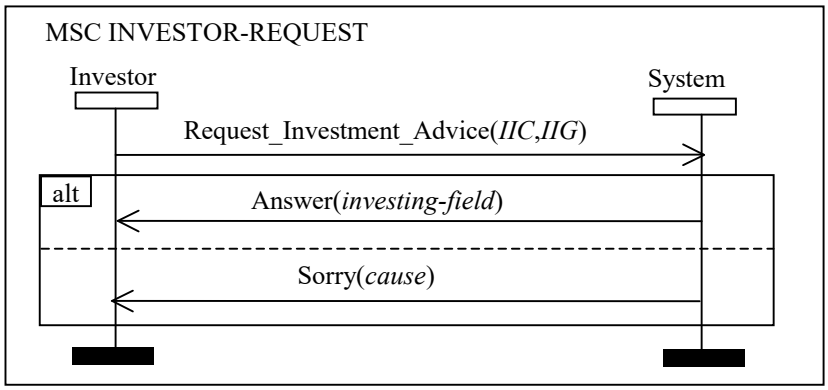

Figure 4.3 MSC Investor request use case diagram

4.3.2 Hybrid Strategy Identification Modelling

The *hybrid strategy identification (HSI) model* consists of the following components: 1) an algorithm to indicate what intelligent techniques to be taken by the system according to the results of hybrid problem requirements; 2) an algorithm to indicate what hybrid strategy is adopted by the system based on the hybrid

problem descriptions and the system's environment; 3) a mechanism to select hybrid technique relations (*COMBINATION*, *TRANSFORMATION*, and *FUSION*); (4) a list of hybrid techniques adopted by the system; 5) a knowledge base to accumulate the knowledge used by the *hybrid strategy identification model*. The relationships of these components are presented in Figure 4.4.

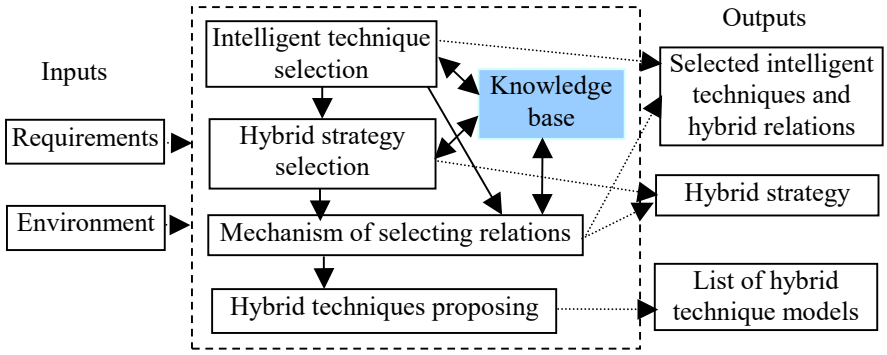

Figure 4.4 Relationships of components in *HSI* model

The inputs of this model are the hybrid problem requirements, environment statements, and knowledge. An environment provides the conditions under which entity exists (Odell, Parunak et al. 2002). The environment consists not only of all the other entities, but also those principle and processes under which the agents exist and communicate. There is a knowledge base in this model, which supports intelligent techniques selection, hybrid strategy selection, hybrid technique relations selection, and the hybrid techniques proposing.

The outputs of this model include: (1) the statements of the selected intelligent techniques and hybrid technique relations; (2) the descriptions of the selected hybrid strategy; (3) the list of hybrid technique models adopted by the system. The *statements of the selected intelligent techniques and hybrid technique relations* will be used by organisation model for helping to decide the categories and groups. The information about hybrid strategies not only affects the agent categorising and grouping, but also helps developers to design the agent architecture and the group architecture. The *descriptions of the selected hybrid strategy* will be used by organisation model and reorganisation model for helping to analyse the agent

categorising, agent grouping, and dynamic rules. The hybrid technique models will be detailed (proposed) in expertise model.

The algorithm to indicate what intelligent techniques to be taken by the system includes the following five steps.

- ***Step 1***: extract the key characteristics of the intelligent processes from the hybrid problem requirements;
- ***Step 2***: use *if – then* rules located in the knowledge base to select suitable intelligent techniques;
- ***Step 3***: calculate the power for each selected intelligent technique;
- ***Step 4***: sort intelligent techniques based on their powers in descent;
- ***Step 5***: select the first *N* or less intelligent techniques in according with the *N* intelligent processes (if the number of the technique candidates is not enough, select all).

The algorithm to indicate what hybrid strategy is adopted by the system includes the following six steps.

- ***Step 1***: extract the key relationships between intelligent processes from the hybrid problem requirements;
- ***Step 2***: extract environment keywords from the environment description of the hybrid problem requirements;
- ***Step3***: use *if – then* rules located in the knowledge base to select suitable hybrid strategies;
- ***Step 4***: calculate the power for each selected hybrid strategy;
- ***Step 5***: sort hybrid strategies based on their powers in descent;
- ***Step 6***: select the first hybrid strategy as the system's main hybrid strategy (*MHS*). If there are more than one hybrid strategies, select the second one as assistant hybrid strategy (*AHS*).

The mechanism to select hybrid technique relations decides which hybrid technique relations are suitable according the selected hybrid strategies and intelligent techniques the system will take. The mechanism includes the following steps:

- ***Step 1***: select the hybrid technique relation(s) (*R1, R2)* using the selected hybrid strategy *MHS* according to the rules (refer to Figure 3.2): transformational→(╠, Θ); loose-coupling→(╠, -); tight-coupling→(╠, -); fully-integration→(╬, -);
- ***Step 2***: if *MHS* is stand-alone, finished with no output; otherwise,
  if *MHS* is transformational, let Z=*T* (refer to Section 3.2.2);
  if *MHS* is loose-coupling, let Z=*LC* (refer to Section 3.2.2);
  if *MHS* is tight-coupling, let Z=*TC* (refer to Section 3.2.2);
  if *MHS* is fully-integration, let Z=*F* (refer to Section 3.2.2);
- ***Step3***: search Table 3.2 using *R1* to find pair or triple intelligent techniques *t*;
- ***Step4***: if $t \in Z$, finished with output *R1*; otherwise, replace *R1* with *R2* in step 3; if $t \in Z$, finished with output *R2*; otherwise, replace *MHS* with *AHS* in step 1 to find the output.

All above knowledge, which is used in this mechanism, described in Section 3.2 is stored in the knowledge base.

4.4 Analysis

The results of this phase are the requirements specification of the agent-based HIS through the development of the models described in Section 4.2, except for the *hybrid strategy identification model* and *design model*. These models are developed in a risk-driven way, and the steps are:

- *Organisation modelling*: developing the organisation model. The construction of these models in the analysis phase is done by means of worksheets.
- *Task modelling*: task decomposition and determination of the goals and ingredients of the tasks.
- *Agent modelling*: developing initial instance of the agent model for identifying and describing the agents.
- *Coordination modelling*: developing the coordination model for describing the interactions and coordination protocols between the agents.

- *Knowledge Modelling*: developing expertise model which includes the knowledge on the domain, the agents (knowledge needed to carry out the tasks and their proactive behaviour) and the environment (beliefs and inferences of the world, including the rest of agents).
- *Reorganisation modelling*: modelling the dynamic feature of agent-based HIS for organising agents with hierarchical structure. In this case, four instances of the reorganisation model are developed: agent category role (each category located in the different level of the hierarchical structure), agent group roles (primitive members of each category), virtual organisation role (the members of a running application), and agent dynamics (coordination mechanism).

4.4.1 Organisation Modelling

The organisation model describes the organisation in a structured, systems-like fashion. The organisation model includes different aspects, such as organisation structure, processes, staff, and resources (Schreiber, Akkermans et al. 1999). The idea is that in the model these components have to be filled in both for the current and the future situation. The inputs of this model are requirements, environment, and the outputs of the *hybrid strategy identification model*. The selected hybrid strategy and the list of the hybrid techniques of the *hybrid strategy identification model* will be processed and delivered to the reorganisation model and the expertise model by this organisation model.

Table 4.2 Worksheet OM-1: Problems and opportunities

Organisation Model	Problems and Opportunities Worksheet OM-1
PROBLEMS AND OPPORTUNITIES	Make a shortlist of perceived problems and opportunities, based on interviews, brainstorm and visioning meetings, discussions with managers, etc.
ORGANISATIONAL CONTEXT	Indicate in a concise manner key features of the wider organisational context, so as to put the listed opportunities and problems into proper perspective.
SOLUTIONS	List possible solutions for the perceived problems and opportunities, as suggested by the interviews and discussions held, and the above features of the organisational context.

The first part of the organisation model focuses on problems and opportunities, as seen in the wider organisational context. The second part contains broad categories, such as, the organisation's mission, goals, strategy, value chain, and external influencing factors. The last part involves the processes categorising and grouping from the hierarchical structure point of view. Table 4.2 gives a worksheet (numbered OM-1) which explains the various aspects to consider, and helps in specifying this part of the organisation model.

The second part of the organisation model concentrates upon the more specific, so-called variant, aspects of the organisation. Table 4.3 gives a worksheet (numbered OM-2).

Table 4.3 Worksheet OM-2: Description of organisational aspects

Organisation Model	Variant Aspect Worksheet OM-2
STRUCTURE	Give an organisation chart of the considered (part of the) organisation in terms of its departments, groups, units, sections,…
PROCESS	Sketch the layout of the business process at hand. A process is the relevant part of the value chain that is focused upon. A process is decomposed into tasks.
PEOPLE	Indicate which staff members are involved, as actors or stakeholders, including decision makers, providers, users or beneficiaries. These people do not need to be actual people, but can be functional roles played by people in the organisation.
RESOURCES	Describe the resources that are utilised for the business process. These may cover different types.
KNOWLEDGE	Knowledge represents a special resource exploited in a business process.
CULTURE & POWER	Pay attention to the unwritten rules of the game, including styles of working and communicating, related social and interpersonal skills, and formal as well as informal relationships and networks.

The last part of the organisation model focuses on the categories, the primitive members (groups) of each category, and the organisations. This part includes the following components: the structure of the process categories and the primitive members, the rules to categorise and group the processes, and documents to express the organisations. The structure of the process categories and the primitive members of agent-based HIS is shown in Figure 4.5. Some processes in different categories and groups can be organised together as a virtual organisation (VO) in MAHIS. The

processes in a virtual organisation comprise an application (also called subsystem) in agent-based HIS.

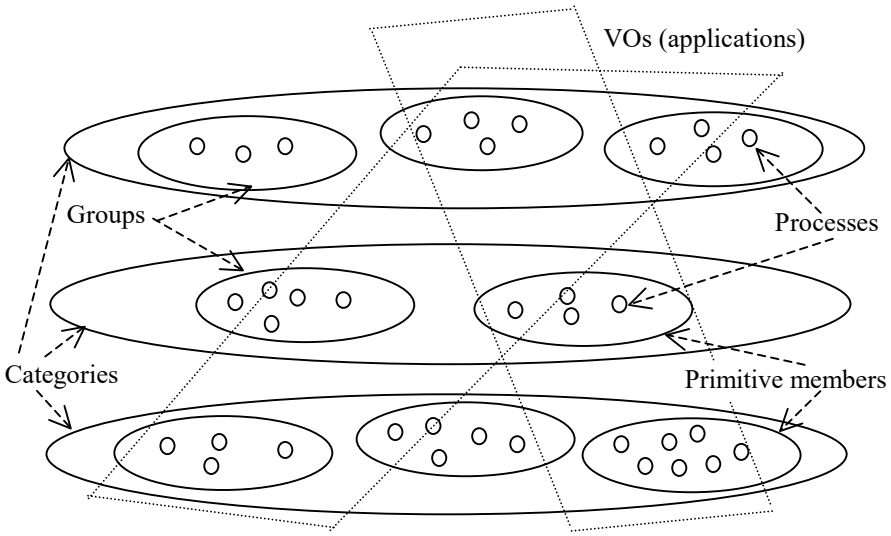

Figure 4.5 The structure of the process categories

The rules to categorise and group the processes are expressed in the following steps:

Step 1: decide the process categories according to the selected hybrid strategy;

Step 2: decide the primitive members of each category according to the selected intelligent techniques and hybrid technique relations;

Step 3: describe the dynamics of the primitive members in each category.

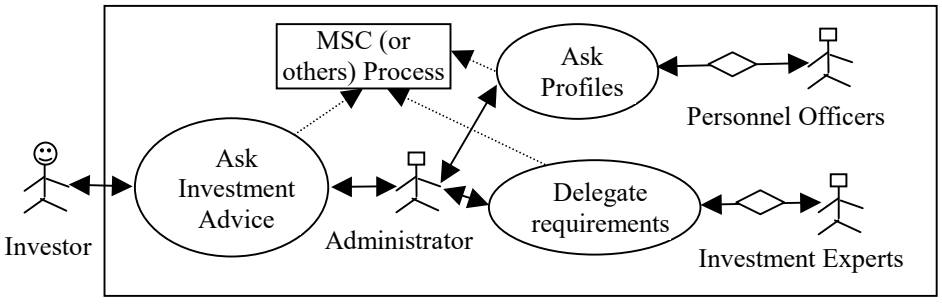

Figure 4.6 The organisation of financial investment planning

The organisation documents include organisation use case, agent categorising information, agent grouping information, and the description of the primitive

member dynamics. Because the initial information for categorising and grouping agents is very important in the reorganisation model, the category and group information must be supplied in this model. The organisation, where the processes and applications are located, must be described for the identification of agent categories and groups. The organisation is represented in the use case notation of OOSE. The organisation of the financial investment planning system is shown in Figure 4.6 as an example.

Four kinds of graphical models are used to develop the organisation model: use case notation of OOSE for representing the organisation, MSC, packages of AUML, templates of AUML, or protocol diagrams for depicting the processes, DFD (Data-Flow Diagram) for describing the relationships between processes and resources, and HRD (Hierarchical Relationship Diagram which is similar to Figure 4.5; note: the oval can be replaced by rectangle for convenience) or extended deployment diagram of AUML (Odell, Parunak et al. 2000) for depicting the results of categorising and grouping.

4.4.2 Task Modelling

According to the definition of task by Schreiber, Akkermans, et al. (Schreiber, Akkermans et al. 1999), a task is a subpart of a business process that: 1) represents a goal-oriented activity adding value to the organisation; 2) handles inputs and delivers desired outputs in a structured and controlled way; 3) consumes resources; 4) requires and provides knowledge and other competence; 5) is carried out according to given quality and performance criteria; 6) is performed by responsible and accountable agents. Following this definition of what a task is, the information covered in the task model is specified with the help of worksheet TM-1, given in Table 4.4.

Tasks are decomposed following a top-down approach, and described in an "and/or tree". The description of a task includes its name, a short description, input and output ingredients, task structure, its control, frequency of application, preconditions and required capabilities of the performers. Some of the items in the

task model, such as value, quality, and performance, refer directly to organisational considerations. Other items in the task model, notably dependency/flow, and time/control, have a natural link with other approaches to information-systems modelling.

Table 4.4 Worksheet TM-1: Refined description of the tasks

Task Model	Task Analysis Worksheet TM-1
TASK	Task identifier and task name
ORGANISATION	Indicate the process this task is a part of, and where in the organisation it is carried out
GOAL AND VALUE	Describe the goal of the task and the value that its execution adds to the process this task is a part of
DEPENDENCY AND FLOW	Input tasks: tasks delivering inputs to this task Output tasks: tasks that use the outputs of this task Use a *data-flow diagram* here to describe this.
TIMING AND CONTROL	Describe frequency and duration of the task. Describe the control relation with other tasks (*State chart*).
AGENT	The staff members or software systems that are responsible for carrying out the task.
KNOWLEDGE AND COMPETENCE	Competences needed for successful task performance. To indicate which elements of the task is knowledge intensive.
RESOURCES	Describe and preferable quantify the various resources consumed by the task.
QUALITY AND PERFORMANCE	List the quality and performance measures that are used by the organisation to determine successful task execution.

Three kinds of graphical models are used to develop the task model: And/or tree for describing the tasks, DFD for representing the relationships between tasks, and state chart of AUML (Odell, Parunak et al. 2000) for describing the control relation with other tasks.

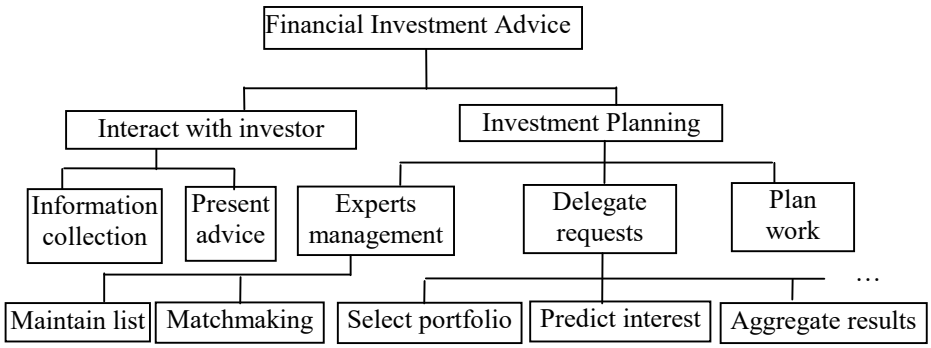

Figure 4.7 The task tree of finical investment planning

In the proposed example presented in Figure 4.6, the task tree is presented in Figure 4.7.

4.4.3 Agent Modelling

The purpose of the agent model is to understand the roles and competences that the various actors in the organisation bring with them to perform a shared task. The information contained in the agent specification is for a large part a rearrangement of information already existing in previous worksheets. The agent has five attributes: name, type (human, new system agent or predefined system agent), role, position, category, and groups (agent groups the agent belongs to). Other constituents of the agent model are service, goal, reasoning capabilities, general capabilities, constraints, etc. Service is the facilities offered to the rest of agents to satisfy their goals. It can perform one task of the task model, and has five attributes: name, type, task, and ingredients. Goal is the objectives of the agents. The goal has the following attributes: name, description, type and ingredients. The goal can be satisfied according to the reasoning capabilities of the agent. Reasoning capabilities are the requirements on the agent's expertise imposed by the task assignment. These are realized by the expertise model. General capabilities are the skills (sensors and effectors to manipulate the environment) and languages the agent understands (agent communication language and knowledge representation language). Constraints are norms, preferences and permissions of the agent. The norms and preferences have special interest in the case of agent-based HIS.

Agents can be identified with the following strategies (or a combination of them) (Iglesias, Garijo et al. 1997):

- ✓ Analysis of the actors of the use cases defined in the conceptualisation phase based on the descriptions of organisation model and task model. The actors of the use cases and the tasks described in task model delimit the external agents of the system. Several similar roles (actors) can be mapped onto one agent to simplify the communication.

- ✓ Usage of heuristics. The agents can be identified determining whether there is some conceptual distance: knowledge distribution, geographical distribution, logical distribution or organisational distribution.
- ✓ The task and expertise models can help us to identify the necessary functions and the required knowledge capabilities, resulting in a preliminary definition of the agents. The goals of the tasks will be assigned to the agents.
- ✓ Application of the internal use cases technique. Taking as input the use cases of the conceptualisation phase and some initial agents, we can think that each agent uses other agents, and can use these agents with different roles. The use case notation is extended for showing human agents (with the round head) and soft agents (with the squared head). When an agent needs to use an agent for a particular function, such an agent is looked for in the agent-library for reusing, combining in this way the top-down and bottom-up approach.
- ✓ The intelligent techniques and hybrid technique models. The intelligent techniques and hybrid technique models are important processes in agent-based HIS. They can be easily identified by following the outputs of the hybrid strategy identification model.

When the agents are identified, the textual template of the agent model should be filled in for each agent that includes its name, type, role, position, groups, a description, offered services, goals, skills, reasoning capabilities, general capabilities, norms, preferences, and permissions.

The *activity diagram* or *state chart* of the AUML (Odell, Parunak et al. 2000) can be used to develop the agent model. Specification of an agent protocol requires spelling out the detailed processing that takes place within an agent in order to implement the protocol. The state charts and activity diagrams can specify the internal processing of agents that are not aggregated.

In the proposed example presented in Figure 4.6, the agents can be identified according to the aforementioned strategies. The administrator's behaviour falls into two tasks: *interaction with user* (*Interface Agent*) and *work planning* (*Planning Agent*). The personnel officer's behaviour falls into two tasks: *expert profiles*

maintenance and *expert matchmaking* (*Middle Agent*). The experts' behaviour falls into five tasks: *financial risk tolerance assessment* (*Financial Risk Tolerance Assessment Agent*), *asset allocation* (*Asset Allocation Agent*), *portfolio selection* (*Portfolio Selection Agents*), *interest prediction* (*Interest Prediction Agents*), and result aggregation (*Decision Aggregation Agent*). Figure 4.8 shows the planning processing behaviour for planning agent in *activity diagram*. Figure 4.9 shows it in *state chart*.

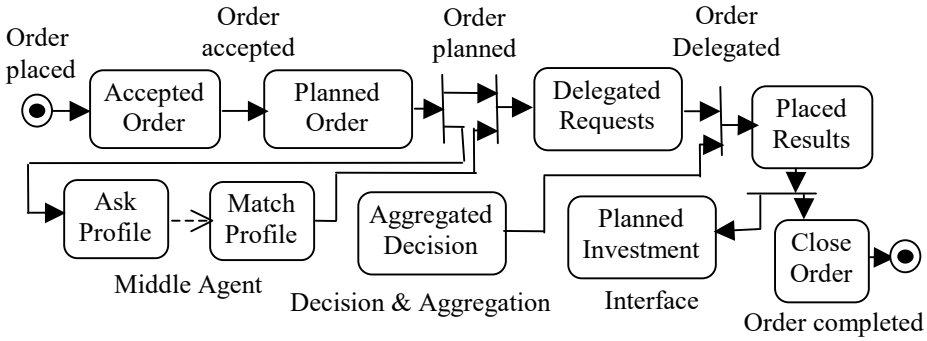

Figure 4.8 Activity diagram for planning agent

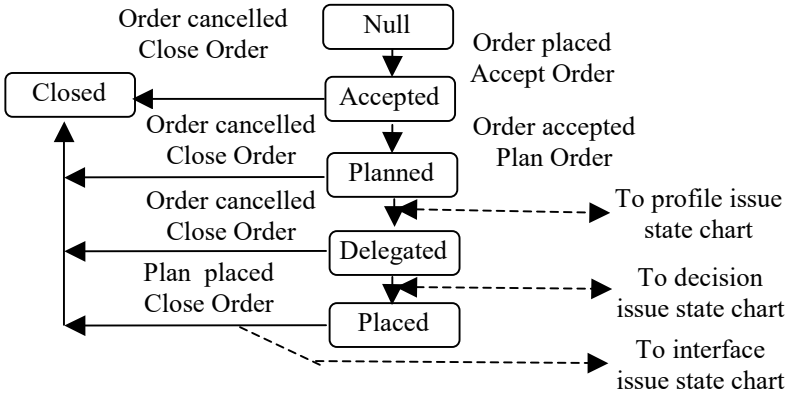

Figure 4.9 State chart for planning agent

4.4.4 Coordination Modelling

The coordination model contains four constitutes (Iglesias, Garijo et al. 1996): conversation, interaction, capabilities, and protocols as shown in Figure 4.10. The

conversation is a set of interactions in order to ask for a service or request or update information. It is distinguished by the name and the requested service name. The interaction is a simple interchange of messages. It has the following attributes: speech-act, agent communication language, knowledge representation language, synchronization, transmitter, receiver and ingredients. The capabilities are the skills and knowledge of the initiator of the conversation and the other participants. The protocol is a set of rules that govern the conversation. It defines the different states and interactions allowed.

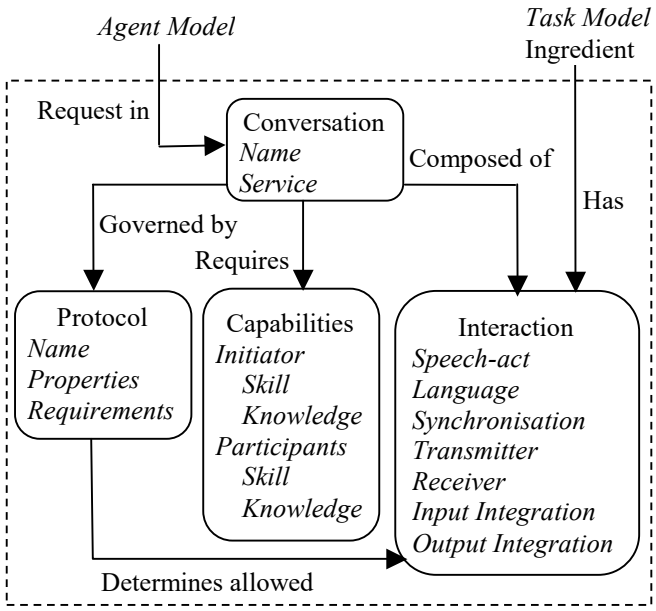

Figure 4.10 Coordination model

The coordination model has two milestones (Iglesias, Garijo et al. 1997): 1) definition of the communication channels and building of a prototype; 2) analysis of the interactions and determination of complex interactions.

The first phase consists of the following steps:

1. Describe the prototypical scenarios between agents using MSC, sequence diagram of AUML, or collaboration diagram of AMUL (Odell, Parunak et al. 2001). The conversations are identified taking as an input the results of the techniques used for identifying agents. During this first stage, we will consider

that every conversation consists of just one single interaction and the possible answer.

2. Represent the events (interchanged messages) between agents in event flow diagrams (also called service charts) (Odell, Parunak et al. 2000). These diagrams collect the relationships between the agents *via* services. Figure 4.11 presents the events of planning agents in event diagram.

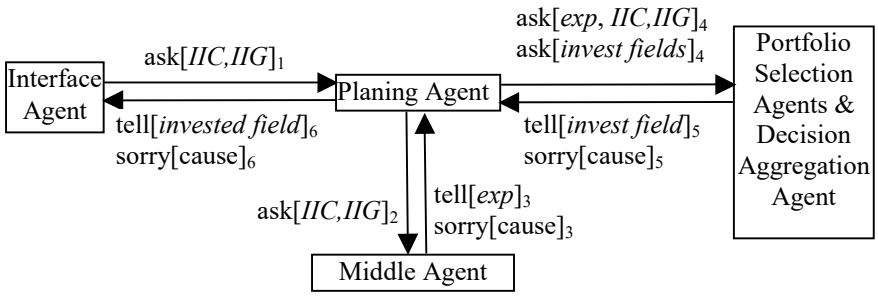

Figure 4.11 The events of planning agents

3. Model the data interchanged in each interaction. The expertise model can help us to define the interchanged knowledge structures. These interchanged data are shown in the event flow diagram between squared brackets.
4. Model each interaction with the state transition diagrams of SDL (Specification and Description Language) (Turner 1993) specifying speech-acts as inputs/outputs of message events. These diagrams can be validated with the MSC diagrams (or other equivalent diagrams). Figure 4.12 presents the interactions of financial investment planning in the state transition diagram.
5. Each state can be further refined in the task or expertise model.
6. Analyse each interaction and determine its synchronisation type: synchronous, asynchronous or future.

The second phase consists of analysing the interactions for getting more flexibility (relaxing for example the user requirements), taking advantage of the parallelism, duplicating tasks using different methods or resolving detected conflicts. When a cooperation protocol is needed, we should consult the library of cooperation protocols and reuse a protocol definition. If there is no protocol suitable for our needs, it is necessary to define a new one. We can use HMSC (High level

Message Sequence Charts) (Turner 1993), which are very useful for this purpose. These diagrams show the road map of the protocol, and how the different phases specified with MSC (or other equivalent diagrams) are combined. A phase can be a simple MSC or another HMSC. The processing of the interactions is described using AUML state chart, and it is also necessary to fill in the textual protocol template specifying the required reasoning capabilities of the participants in the protocol. These capabilities can be described using one or several instances of the expertise model. The state charts consider three kinds of events: message events, events from other agents using message-passing; external events, events from the environment perceived through the sensors; and internal events, events that arise in an agent because of its proactive attitude.

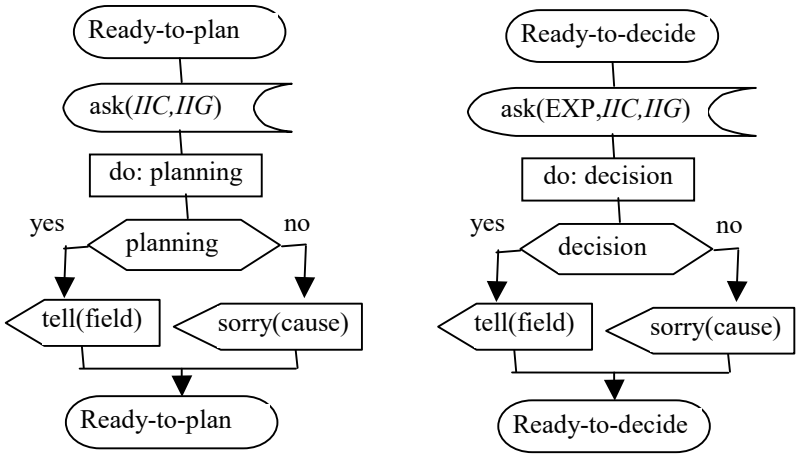

Figure 4.12 Interactions with SDL state diagrams

4.4.5 Knowledge Modelling

The expertise model is used for modelling the reasoning capabilities of agents to carry out their tasks and achieve their goals. Normally, several instances of the expertise model should be developed: modelling inferences on the domain; modelling the reasoning of the agent (i.e. problem solving methods to achieve a task, character of the agent, etc.), and modelling the inferences of the environment (how

an agent can interpret the event it receives from other agents or from the world). When we have to develop the reasoning capabilities of an agent, we will reuse previously developed instances of the expertise model and adapt these instances to the new characteristics of the problem (Studer, Benjamings et al. 1998).

The expertise model consists of the development of the application knowledge (consisting of domain knowledge, inference knowledge and task knowledge) and problem solving knowledge. The expertise model can be defined as following by using CML (Conceptual Modelling Language) (Schreiber, Akkermans et al. 1999):

expertise-model ::= **knowledge-model** *Expertise-model;*
[terminology] /* Zero or one occurrence */
domain-knowledge
inference-knowledge
task-knowledge
[psm-knowledge]
end knowledge-model[*Expertise-model;*].

It is usual for psm (problem solving method) knowledge to be defined separately, for example as part of a library of PSMs.

Domain Knowledge: represents the declarative knowledge of the problem, modelled as concepts, properties, expressions and relationships using the CML or the graphical notation of the Object Model of OMT. The domain knowledge can be defined by the CML as following:

domain-knowledge ::= **domain-knowledge** *Domain-knowledge;*
[terminology]
<domain-schema |
ontology-mapping | knowledge-base>*
end domain-knowledge[*Domain-knowledge*;].

domain-schema ::= **domain-schema** *Domain-schema;*
[terminology]
[**use** : use-construct, …;]
[**definitions** :] domain-struct*
end domain-schema [*Domain-schema*;].

use-construct ::= *Domain-schema* | *Construct* from *Domain-schema*.

domain-construct ::= binary-relation | concept | mathematical-model |
relation | rule-type | value-type.

ontology-mapping ::= **ontology-mapping** *Ontology-mapping;*
[terminology]
from : *Domain-schema* ;
to : *Domain-schema* ;
mappings : "Text"
end ontology-mapping [*Ontology-mapping*;].

knowledge-base ::= **knowledge-base** *Knowledge-base;*
[terminology]
use : knowledge-base-use , … ;
[[**instances** :] <instance | tuple > $^{+}$]
[**variables** : variable-declaration ; … ;]
[**expressions** : knowledge-base-expression ; … ;]
[**annotations** : "Text" ;]
[attributes]
end knowledge-base [*Knowledge-base*;].

Inference Knowledge: represents the inference steps performed for solving a task. There is a library of generic inference structures selected by the task type (diagnosis, assessment, etc.). These generic inference structures should be adapted to the problem. The inference structure is a compound of predefined inference types and domain roles. After defining the inference structure, it is instantiated into a similar diagram for the domain. The inference knowledge can be defined by the CML as following:

inference-knowledge ::= **inference-knowledge** *Inference-knowledge;*
[terminology]
[**use** : use-construct, …;]
<inference |
knowledge-role | trandfer-function>*
end inference-knowledge[*Inference-knowledge*;].

Task Knowledge: represents the order of the inference structures. The notation consists of inference structures and task-method inference decomposition structures. The task knowledge can be defined by the CML as following:

task-knowledge ::= **task-knowledge** *Task-knowledge;*
[terminology]
[**use** : *Inference-knowledge* , …;]
task-element*

end task-knowledge[*Task-knowledge*;].

Problem Solving Method: during the design, a Problem Solving Method (PSM) should be specified for each inference type: how the inference is carried out. The PSMs are arranged in libraries for reuse. The PSM knowledge can be defined by the CML as following:

psm-knowledge ::= **psm-knowledge** *Psm-knowledge;*
[terminology]
psm-description*
end psm-knowledge[*Psm-knowledge*;].

The conventions used in the CML syntax specification are listed in Table 4.5 (Schreiber, Akkermans et al. 1999).

Table 4.5 Conventions for CML syntax specification

X ::= Y	The syntax of X (a non-terminal) is defined Y.
[X]	Zero or one occurrence of X.
X*	Zero or more occurrence of X.
X+	One or more occurrence of X.
X Y …	One or more occurrence of X separated by Y.
X | Y	One of X or Y (exclusive or).
<X>	Grouping construct for specifying of the scope of operators.
symbol	Bold: predefined terminal symbols of the language.
Symbol	Capitalised: user-defined terminal symbols of the language.
symbol	Lowercase: non-terminal symbols.
"Text"	Arbitrary text between double quotes.
"X"	Escapes the operator symbol (e.g., *) and denotes the literal X.

4.4.6 Reorganisation Modelling

Reorganisation model is developed to describe the dynamic feature of HIS with virtual organisation (VO) (Norman, Preece et al. 2003), category, and group perspectives (Parunak and Odell 2002). This model shows the organisational relationships between agents for supporting dynamic addition and removal of agents in agent-based HIS. At the same time, this model has the ability to make the agents

that are developed previously reusable. Because each agent can be dynamically involved in a new VO, the agents in the system are reorganised.

Reorganisation model consists of four components: category role, group roles, VO role, and dynamics rules. The relationships between those components are shown in Figure 4.13.

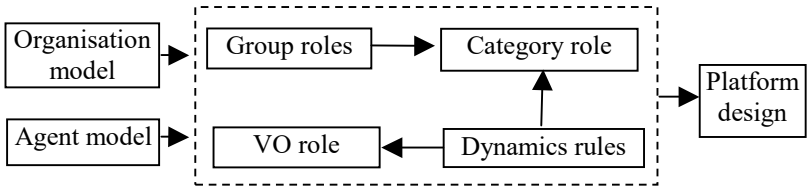

Figure 4.13 The components of reorganisation model

In agent-based HIS, agents are divided into several categories. Each category locates in a specific level in a hierarchical structure. The category role indicates those categories. A category is an instance of the category role. The following questions must be answered in the category role. How many categories must be divided? What is the type of members in each category like? What is the hierarchy between agent categories? What are the characteristics of the organisational structures organised the members for each category? The description of category role includes instances (categories), member type (one of the group roles), hierarchy, and structure type. However, it is conceptually wrong to think of a structure type as something that actually defines the organisational structure (Zambonelli, Jennings et al. 2000). Instead, in the design phase, the structure type should derive from the organisational structure that is explicitly chosen. Table 4.6 gives a worksheet (numbered RM-1). The category role can be depicted in AUML class diagram (Parunak and Odell 2002).

For the purposes of dynamic addition and removal of agents, a member in a category may consist of more than one agent. The member of each category is an agent group, which is described in group role. An agent group is an instance of group role. Each category has its own group role. So the number of the group roles is not more than the number of categories. The agents in a group can coordinate to achieve their goal or they have closer relationships. Agent group roles describe the

aforementioned features of an agent-based system. The following attributes must be described in each group role. What are the rules for grouping the agents? What category does the group role belong to? What is the organisational structure adapted in a group for organising the agents? Are the agents in a group reusable? The description of each group role includes group role name, rules for grouping agents, instances of the group role, category belonged to, structure type, and reusability of the agents. Table 4.7 gives a worksheet (numbered RM-2). The group roles can be depicted in AUML class diagram.

Table 4.6 Worksheet RM-1: Description of category role

Reorganisation Model	Category Role Worksheet RM-1
CATEGORIES	Make a shortlist of possible categories. Give each category an identifying name. The result can be presented in a two-column table (category name, description of what kind of agents)
MEMBER	Indicate member type of each category. The member type is one of the group roles. The result can be presented in two-column table (category name, group role name).
HIERACHY	Indicate the relationships between categories. Because the purpose to categorise agents is to manage them in hierarchy, there is hierarchical relationship among those categories.
STRUCTURE TYPE	Indicate the types of organisational structures for all members in each category. The results can be presented in context and are used by the platform design phase.

Table 4.7 Worksheet RM-2: Description of agent grouping

Reorganisation Model	Agent Grouping Rules Worksheet RM-2
GROUP ROLE NAME	Present the group role name associated a category and the description of the group role.
RULES	Describe the rules to select the agents in a category into groups. Usually agents are grouped according to their roles in achieving their goal.
CATEGORY	Indicate that the group role belong to which category.
GROUPS	Make a shortlist of possible groups. Give each group an identifying name. The result can be presented in a two-column table (group name, description of agents)
REUSABLE	Indicate if the agents in a group are reusable. If the agents can be reused in a group, they may be organised in a VO.
STRUCTURE TYPE	Indicate the types of organisational structures for all agents in each group. The results can be presented in context and are used by the platform design phase.

For reusing agent, an instance (so-called virtual organisation) of the VO role may consist of more than one agent across all or some categories. A VO can be regarded

as a subsystem or an application in the agent-based HIS. An agent may belong to more than one VO at the same time. When a VO registered into the system, its member may change dynamically. A VO may include all agents of a group or parts of its agents. The description of the VO role includes instances (VOs), VO member, and structure type. The VO roles can be depicted in AUML class diagram. Figure 4.14 shows relationships between category, group, VO, and agent using AUML class diagram.

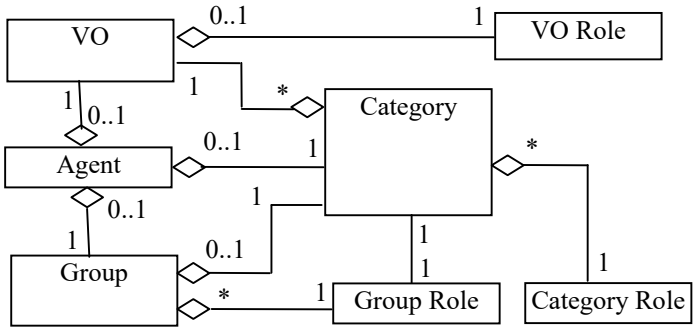

Figure 4.14 Category, group, VO, and agent relationships

In presenting an overall picture of the VO management process, we will use a specific scenario (Norman, Preece et al. 2003). Firstly, there may be multiple available services from a number of agents, each of which locates in a specific group. For example, portfolio selection agents in financial investment planning: SP1 (Markowitz's model), SP2 (fuzzy probability portfolio selection model), and SP3 (possibility portfolio selection mode). The services themselves are described by multiple attributes, e.g., price, quality, and delivery time. The available services may change over time: new services may become available, or agents may alter the way in which existing services are offered. Services may differ in terms of the number and heterogeneity of the tasks involved in the delivery of the service and their degree of interdependence, and the type and frequency of interactions between different customers while the service is being delivered. The agents involved in the system may also employ different policies for dealing with the uncertainty inherent in such a domain.

Secondly, it is assumed that each service provider advertises the services that they offer to a *middle agent* (MA). This agent is consulted by the *requirements*

agent (RA) and asked to recommend agents that have advertised the ability to solve a complex problem. Thirdly, following the receipt of this information, the *requirements agent* will distribute a call for bids to fulfil a specific set of requirements. Fourthly, the service provider agents must now decide whether and what to bid in response to this call. Lastly, the requirements agent must select some combination of services that best suits the needs of the user. The process of the VO formation is presented in Figure 4.15.

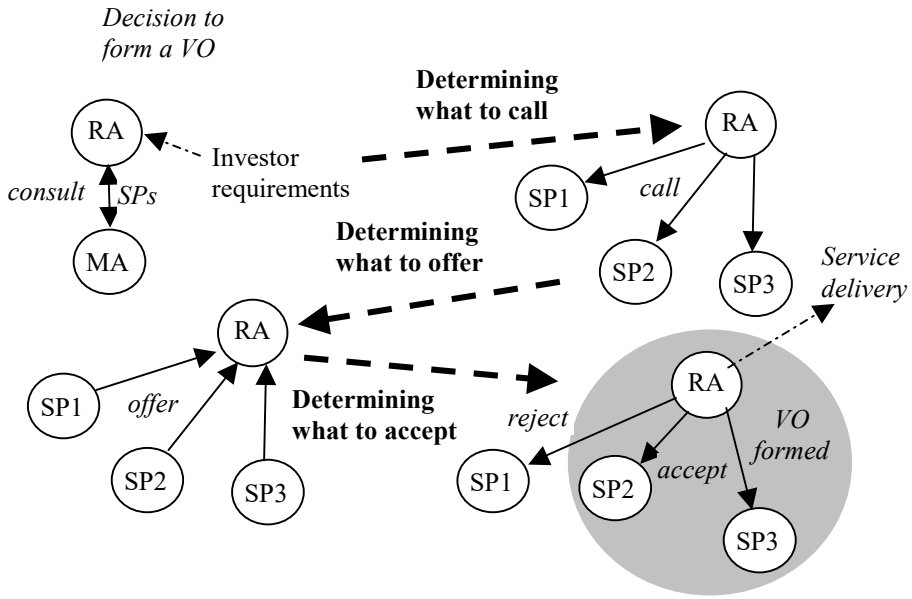

Figure 4.15 The process of VO formation

Dynamics rules include two aspects of rules. The first aspect of rules is about the interactions between categories. These rules indicate the coordination mechanism of the category role. The second aspect of rules is about the operation mechanism of the VO. The dynamic ability of agent-based system is described in system dynamics rules. The following attributes must be described in dynamics rules. What is the coordination mechanism between categories? How to establish and maintain the mechanism for the dynamic addition and removal of groups in each category? What are the rules for reorganising agents as a VO? How or when initialize the agent-based HIS? The description of the dynamics rules includes category mechanism, group mechanism, reorganisation mechanism, and initialization mechanism. Table

4.8 gives a worksheet (numbered RM-3). The dynamics rules can be depicted using MSC, sequence diagram of AUML, or collaboration diagram of AMUL.

Table 4.8 Worksheet RM-3: Description of system dynamics

Reorganisation Model	System Dynamics Rules Worksheet RM-3
CATEGORY MECHANISM	Describe the coordination mechanism between categories. For example, matchmaker, broker, contract-net token ring. The result can be presented in MSC or other equivalents.
GROUP MEACHANISM	Describe the mechanism of dynamic addition and removal of groups in each category. The result can be presented in MSC or other equivalents.
REORGANIZING MECHANISM	Indicate the mechanism to organize agents or groups as a VO. The VO is dynamically managed. The result can be presented in MSC or other equivalents.
INITIALIZATION MECHANISM	Indicate the mechanism and conditions to initialize agent-based system. This process includes system, category, group, VO, and agent initializations.

4.5 Design

We look at the problem of turning requirements specified in the analysis models into implementation specifications in this section. The major inputs for the design process in MAHIS are the expertise, coordination, and reorganisation models. As a result of the analysis phase, the initial sets of agents, groups, categories, and VOs have been determined. The interactions between agents have been clarified by synthesising relevant interaction mechanisms in coordination and reorganisation models. During design phase the design model is developed. A typical design process starts with a specification of the software architecture (Schreiber, Akkermans et al. 1999). Once the general structure of the software is defined though the architecture, a detailed architecture specification can be made. This serves as the basis for the actual platform and VO-based application. In addition, design agents need to be taken with respect to the ACL. According the application, the practical ACL can be adopted or improved. If necessary, a new ACL can be proposed.

This phase consist of four steps.

- **Step1: Architecture design ----** The general architecture of the system is specified according to the structure and mechanism description in reorganisation and coordination models. Because the hierarchical structure of agent categories and the organisational structures of agent groups and agents have been predefined in reorganisation model, this step can therefore be carried out quickly.
- **Step 2: Agent communication language** ---- As we have known, KQML (Finin, Labrou et al. 1997) is a popular ACL for transmitting messages between agents. It can be adopted in the agent-based systems. However, for hybrid intelligent systems it is not suitable to transmit special information. According to the application, the practical ACL should be improved or enriched. ACL design is an important task which should be taken before design the platform and any application.
- **Step 3: Platform design** ---- The platform which supports the dynamic addition and removal of agents is designed in this step. When the platform is designed, hardware and software environment should be considered for the convenience of system implementation. When the platform is designed, the all specifications described in reorganisation model must be taken into account.
- **Step 4: Application design** ---- In the final step we take the ingredients from the analysis models (organisation model, task model, agent model, coordination model, reorganisation model) and map those onto the architecture and platform.

4.5.1 Architecture Design

The architectures of MAS can be divided into two levels: system level and agent level. A system architecture description typically consists of three elements (Sommerville 1995): a decomposition of the system into subsystem, the overall control regimen, and the decomposition of subsystems into software agents. About the agent architectures, Wooldridge and Jennings have grouped them into: deliberative architecture, reactive architecture, and hybrid architecture (Wooldridge

and Jennings 1995). Each agent in HIS can adopt one of these agent architectures. In MAHIS, the architecture adopted by each agent is not constrained.

About the system level, MAHIS has defined a reference architecture that can be used in most HIS applications. This architecture is described at three levels of granularity.

- ***Global system architecture***

In this architecture three major subsystems are distinguished: application model, views, and controller. The application model specifies the functions and data that together deliver the functionality of the application. In the agent-based HIS, the application model contains the grouped problem solving agents. The "data" in the application model are the respective knowledge bases and the dynamic data manipulated during problem solving. The "views" specifies external views on the application functions and data. In the agent-based HIS, the views are the concepts of category and VO. The controller is the central "command and control" unit. In the agent-based HIS, the controller is the groups with middle agents. Figure 4.16 shows an example of the global system architecture.

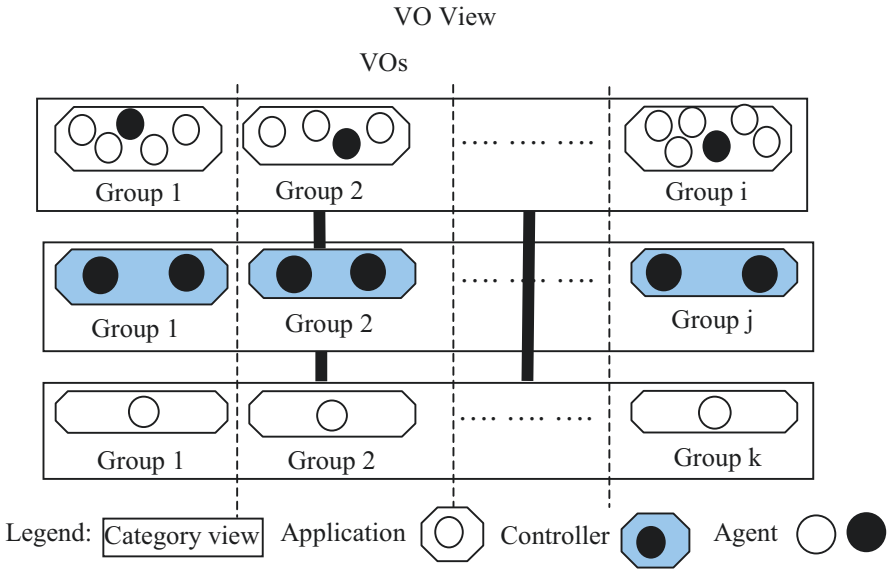

Figure 4.16 The global system architecture

- ***Architecture of the view subsystem***

The view subsystem contains category and VO. In MAHIS, architecture of the view subsystem consists of the components which indicate the organisational relationships of all groups in each category or all agents in each VO.

- ***Architecture of the application model subsystem***

The application model contains the software agents that realise the functions and data specified during analysis. In MAHIS, architecture of the application model consists of the components which indicate the organisational relationships of all agents in each group.

4.5.2 Agent Communication Language

Speech act theories have directly informed and influenced a number of languages that have been developed specifically for agent communication (Wooldridge 2001). KQML is an "outer" language for agent communication. It defines an "envelope" format for messages, using which an agent can explicitly state the intended illocutionary force of message. KQML is not concerned with the *content* part of massages. If KQML does not cover the developers' requirements, the *content* parameter can be defined with the specific performatives of the developed system.

4.5.3 Platform Design

The meaning of platform design here is to construct a dynamic platform rather than the implementation platform identification (Iglesias, Garijo et al. 1996). The platform organises all agents dynamically from the group, category, and VO points of view. The platform design consists of organisational structure modelling and coordination mechanism design.

The dynamic platform organisational structure modelling can derive from the specifications resulted in the analysis phase and architecture design. The organisational structure modelling assigns a topological structure for each category.

The results can be presented in topological graph. There is at least one topological graph for each category.

Coordination mechanism is another factor related to the interaction and organisation among agents, groups, categories, and VOs. Without coordination, agents will not work in order and cannot correctly response to service requests because of the conflicts between them. The coordination mechanism includes three aspects of interactions which can be depicted in MSC or other equivalents. Firstly, the communication links that exit between agent groups must be described. Secondly, the dynamic addition or removal of agent groups should be indicated. Finally, the mechanism and conditions to initialise the platform must be laid out. This process includes system initialisation, category initialisation, VO initialisation, and group initialisation.

4.5.4 Application Design

In application design process we define the application model in more detail. The generic architectural facilities for application model can be provided. The application design consists of agent design, knowledge base design, organisational structure modelling, and group integration. Agent design needs to define two operations: (1) the problem solving operation that can solve a specific problem with a method, and (2) the interaction operation, which can manage to interact with other agents.

For the knowledge bases three decisions have to be taken: (1) we have to decide on the representational format for the instance of rule types; (2) we have to define some access and modify functions; (3) we are likely to need knowledge-base modification and analyse functions.

The dynamic platform organisational structure modelling can derive from the specifications resulted in the analysis phase and architecture design. The organisational structure modelling assigns a topological structure for the groups in each category. The results can be presented in topological graph. The group

integration defines the operation to support the dynamic addition and removal of a group.

4.6 Summary

The methodology MAHIS has been proposed based on MAS-CommonKADS. However, MAHIS is a new methodology because it has its own development lifecycle (HSDLC) and a set of complete rules for analysing and designing agent-based HIS rather than MAS-CommonKADS. MAHIS not only has added the reorganisation model, but also improved the organisation model, agent model, and design model, which are suitable for constructing agent-based HIS. MAHIS includes three process stages: conceptualisation, analysis and design. Following MAHIS, the analysis and design of agent-based HIS can be conducted from the user requirements to the implementation specifications. The results of MAHIS can be implemented based on different developing environments.

MAHIS has three distinct characteristics which are not covered by other agent-oriented methodologies. Firstly, MAHIS is suitable for constructing agent-based HIS as well as any open systems with hierarchical structure. Secondly, MAHIS supports the construction of agent-based systems with the ability of reorganisation of agents. Finally, dynamic platform development is taken into account from the methodology point of view. The platform can dynamically organise all agents in a system.

Chapter 5

5 Case Study 1: PAHIS ---- A Platform for Agent-Based HIS

In this chapter, a dynamic platform PAHIS (***P****latform for* ***A****gent-based* ***H****ybrid* ***I****ntelligent* ***S****ystems*) is introduced for verifying the platform construction ability of MAHIS. PAHIS has been developed by three steps: analysis, design, and implementation. The analysis and design steps are in accordance with the related process stages of MAHIS. Before PAHIS is analysed, the user requirements are described (Section 5.1). Four tasks (category role modelling, group roles modelling, VO role modelling, and dynamics rules' modelling) are conducted in PAHIS analysis (Section 5.2). The category role indicates the rules to categorise agents in the system. At the same time, the categories which act as the category role are described. The group roles depict the primitive members of each category and the rules to organise a group. The VO role depicts the rules to form a VO dynamically. The dynamics rules' modelling is to describe the rules which make agents add into or remove from the platform and VO dynamically. The architectural model of PAHIS is discussed in the design phase (Section 5.3). The architectural model includes organisational structure and coordination mechanism. The ring-based architectural model and token ring coordination mechanism are proposed for improving the flexibility and robustness of the platform. PAHIS is implemented by

using C language and Socket technique (Section 5.4). At last, the ring-based structure is evaluated by comparing it with peer-to-peer, tree, and grouping with facilitator (Section 5.5), which are typical organisational structures adopted by agent systems.

5.1 Requirements for Developing PAHIS

For developing PAHIS, the requirements are given in this section. The developed PAHIS can integrate a set of applications, each of which can complete a specific task. There are several shared processes, each of which provides a specific service. The number of the processes may dynamically change at run-time. Applications in the system can call those shared processes. All applications and processes may enter or leave the system at run-time rather than design time. The requirements are summarised as following:

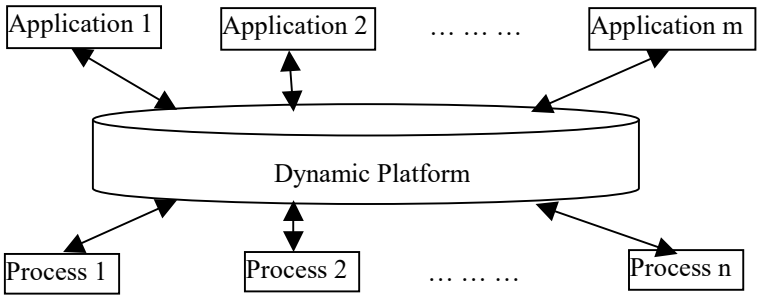

Figure 5.1 The framework of the system

- The processes and applications may be developed with different techniques and environment;
- The components carried out the processes and applications can dynamically leave and enter the system;
- The components carried out the processes and the components located in an application can interact with each other for cooperating to achieve a specific goal;
- There are no self-interested components in the system;

- Integration and interaction of different techniques are crucial;
- The platform is reliable, scalable, and efficient.

Each process is a service provider and each application is a requester. The framework of the developed system is presented in Figure 5.1.

5.2 PAHIS Analysis

According to the requirements descriptions in Section 5.1, we can describe PAHIS using use case notation as shown in Figure 5.2. The interactions between the components are presented in Figure 5.3. An administrator and a yellow page manager are located between application and service provider for matchmaking and managing service providers.

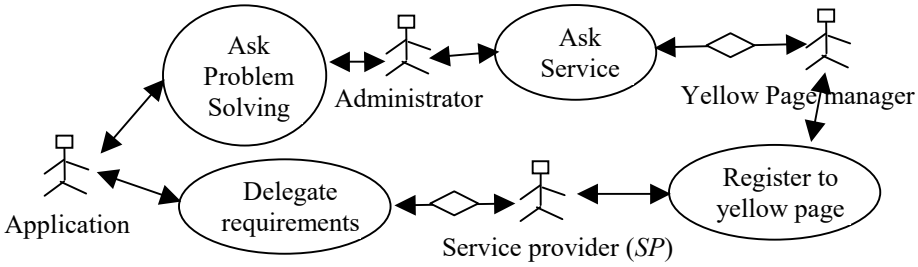

Figure 5.2 The organisation use case of PAHIS

The tasks in this example include: request processing, service provider matchmaking (matchmaking and service provider registering), and problem solving. So there are three kinds of agents in PAHIS in accordance with the tasks. The first kind of agents is application agents in accordance with the request processing. The second kind of agents is middle agents in accordance with the service provider matchmaking. The last kind of agents is service provider agents in accordance with the problem solving. Each kind of agents is located in a category, that is, PAHIS includes three categories as shown in Figure 5.4.

Each application in application agent category is defined as an agent group. There are several agents in each agent group. The agent number of each agent group in this category may not be the same as shown in Figure 5.4. Each group in middle

agent category includes two middle agents (host and it duplicate) and serves for one responsible group in application agent category. The number of groups in the application agent category is the same with the one in the middle agent category. Each group in service provider category just includes one agent.

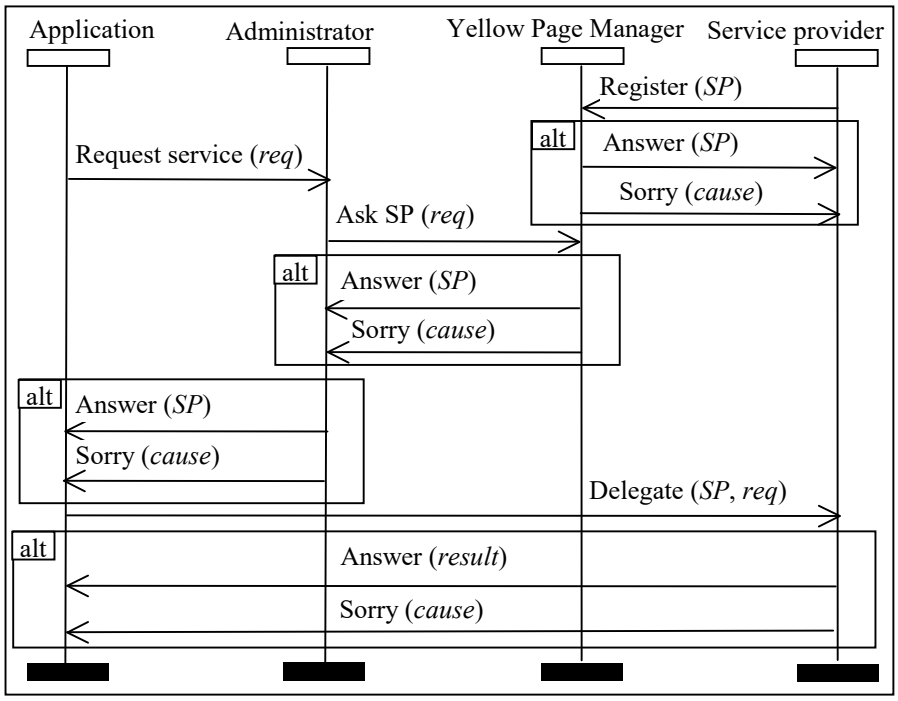

Figure 5.3 MSC interactions of PAHIS

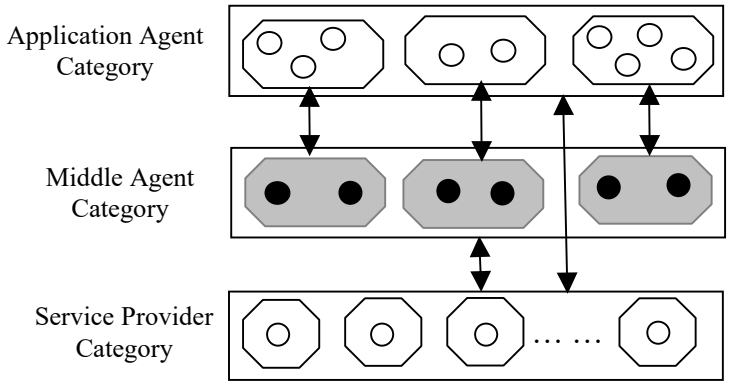

Figure 5.4 The hierarchy structure of the categories

The VO will be automatically defined when an application is run. The members of a VO include all agents in the application group and relevant service provider agents which are called by the running application.

The dynamics rules are described as following:

- When the platform is initialized, there is only one middle agent group waiting for serving an application agent group. If a new application agent group (not the first one) applies to register to the system, a new middle agent group will be added for it. If an application agent group (not the last one) applies to cancel from the system, its middle agent group will be removed. There are two middle agents in each middle agent group for improving the reliability. One middle agent acts as host, and another acts as host's duplicate. If host middle agent crashed, its duplicate would take its position and, at the same time, it would produce a new duplicate. The necessary information of service providers and applications is stored in each middle agent.

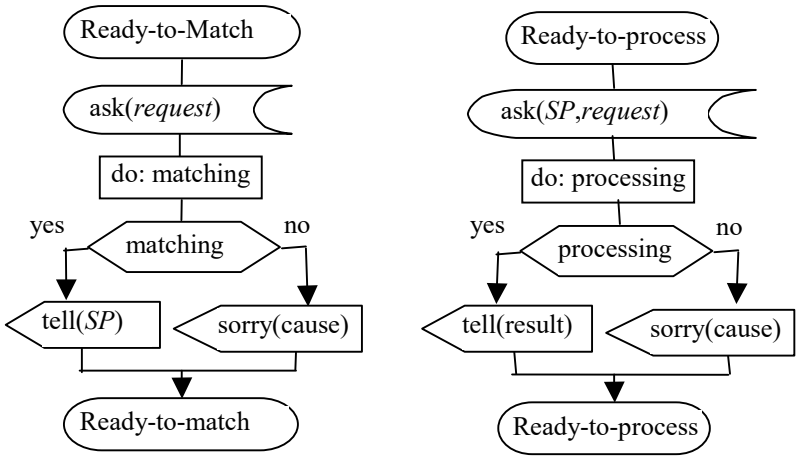

Figure 5.5 SDL interactions of PAHIS

- There is one organising agent in each application group to manage all agents in the application group. This organising agent is in charge of the requests of registration and cancellation of this application group from a specific middle agent. When an application agent group enters the system, its middle agent group will be created and its information will be registered into each middle agent. When an application agent group leaves the system, its middle agent group will be removed and its information will be deleted from each middle agent.

- Each service provider agent can register and cancel its service to the system. When a service provider agent enters the system, its information will be registered into each middle agent. When a service provider agent leaves the system, its information will be deleted from each middle agent.
- Each middle agent has the ability of matchmaking. When an agent in an application agent group needs to get the service from a service provider agent, it requests its service to its middle agent, and the middle agent will tell the requester who is the service agent. Then the requester will contact the service agent directly.

According to the dynamics rules, the main interactions of PAHIS are presented in Figure 5.5 with SDL.

5.3 PAHIS Design

In PAHIS, middle agents are employed for solving the matchmaking problem between service provider agents and requester agents. The performance of middle agents not only relies on the matchmaking algorithms employed by them, but also the architecture that organises them with suitable organisational structure and coordination mechanism (Li, Zhang et al. 2003e). The organisational structure and coordination mechanism of an agent-based system determine the interaction performance among agents. Organisational structure presents the interrelationship among agents in a system, and coordination mechanism is knowledge level protocol to control the sequences of interaction and manage conflicts among agents. According to the role of each agent or interrelationship among agents, an agent-based system has an organisational structure. The organisational structures have been modelled into three types, Peer-to-Peer (P), Tree (T), and Grouping with facilitator (F), in practical agent-based systems. Coordination mechanism is the protocols to manage inter-dependencies between the activities of agents. Coordination mechanisms have been classified into four patterns, Direct search (D), Matchmaker (M), Broker (B), and Contract-net (C). Seven architectural models for

agent-based systems have been proposed by combination of the organisational structures (OS) and coordination mechanisms as shown in Table 5.1 (Li, Zhang et al. 2003e).

Table 5.1 Practical architectural models

OS	D	M	B	C
P	P-D	-	-	P-C
F	-	F-M	F-B	F-C
T	-	-	T-B	T-C

A key issue concerning agent-based systems with middle agents is how to organise requester agents, middle agents, and service provider agents, so that the requester agents can receive appropriate services quickly and efficiently. The current architectural models cannot solve the problem because they do not concentrate on the features of middle agents. Although F-M, F-B, and T-B architectural models are suitable for organising the middle agents by analysing the performance of the seven architectural modes in Table 5.1, there are two fatal limitations (Li, Zhang et al. 2003b). Firstly, the middle agent is becoming the bottleneck between agent interactions of the system. When a requester agent wants to interact with a service provider agent, it must contact middle agent first. If the number of requester agents is enormous, the middle agent will be very busy. At last, because of the overload of the middle agent, the system will be broken down or the efficiency of the system will decrease to a very low point. Secondly, because the reliability of the system absolutely relies on the middle agent, the robustness of the middle agent is the key problem. If the middle agent crashed, the whole system would break down. Certainly, above two limitations can be alleviated by means of increasing the amount of middle agent. Because of the lack of efficient organisational structure, the limitations cannot be fully overcome.

We have contributed a self-organising ring-based architectural model to organise the middle agents in agent-based system. The ring-based architectural model is based on logical ring organisational structure and token ring coordination

mechanism. About the organisational structure, the middle agents are divided into hosts and duplicates according to their roles in the system. About the coordination mechanism, coordinator electing mechanism is employed to build the logical ring, manage token, and produce or cancel middle agents.

5.3.1 The Organisational Structure for Middle Agents

Ring-based architectural model is powerful for improving the scalability and robustness of the key components (e.g., middle agents) of agent-based system. The ring-based architectural model is based on logical ring organisational structure and token ring coordination mechanism. The logical ring organisational structure is shown in Figure 5.6.

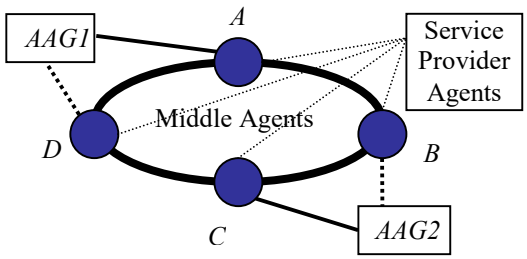

Figure 5.6 Ring organisational structure

The middle agents *A*, *B*, *C*, and *D* serve for requester agents, that is, application agent group 1 (*AAG1*) and application agent group 2 (*AAG2*). The requester agents in an application agent group may be organised as peer-to-peer or tree structure. A requester agent in an application agent group cannot directly contact one in another application agent group. However, a requester agent can belong to more than one application agent group. All information of service provider agents is stored in every middle agent by coordination mechanism. The service provider agent advertises its information to the coordinator of the ring. The information is transmitted to all middle agents by token ring coordination mechanism (see Subsection 5.3.2 for details). Each middle agent stores meta necessary information

(like yellow page) about its supervised child agents to control the interaction among those child agents.

The middle agents are organised as logical ring structure. That is, middle agent *A* can only contact directly middle agent *B* and *D*, but not *C*. If middle agent *A* wants to contact *C*, the message must be transmitted by *B* or *D*. For simplifying control mechanism, the logical ring is designed to be of the feature of one way in transmitting message. For example, middle agent *A* can only directly transmit message to middle agent *B* and receive message from *D*, but it is not true in contrast. The middle agents are divided into two types, namely, host and duplicate, according to their roles in the system. The host middle agents, for example, *A* and *C*, are the superior agents of *AAG1* and *AAG2*, respectively, in normal case. If the host middle agent *A* or *C* crashed, the ring would be automatically reorganised. At the same time, its duplicate middle agent *D* or *B* will act as a host and automatically produce a new duplicate middle agent. The meta information in host middle agent and its duplicate must be synchronised in time.

5.3.2 Coordination Mechanism

The token ring coordination mechanism consists of two algorithms, namely, logical ring establishing algorithm and token control algorithm.

For efficiently establishing the logical ring of middle agents, there is a ***coordinator*** among them at any time to establish a logical ring (system beginning, token lost or the ring broken down). In general, it does not matter which middle agent takes on this special responsibility, but one of them has to do it. For selecting a middle agent as coordinator, each middle agent is assigned a unique identify number (*UIN*; 0 to 255 in the platform). In general, election algorithm attempts to locate the middle agent with the highest *UIN* and designate it as coordinator. Furthermore, it is assumed that every middle agent knows the *UIN* of every other middle agent. What the middle agents do not know is which ones are currently up and which ones are currently down. The goal of election algorithm is to ensure that

when an election starts, it concludes with all middle agents agreeing on who the new coordinator is to be. In our research, we employed ring election algorithm (Tanenbaum 1995). This algorithm is based on the use of a ring, but without a token. It is assumed that the middle agents are logically ordered, so that each middle agent knows whom its successor is. When any middle agent notices that the coordinator is not functioning (the token interval is overtime), it builds an ELECTION message containing its own *UIN* and sends the message to its successor. If the successor is down, the sender skips over the successor and goes to the next member along the ring, or the one after that, until a running middle agent is located. At each step, the sender adds its own *UIN* to the list in the message. Eventually, the message gets back to the middle agent that started it all. That middle agent recognises this event when it receives an incoming message containing its own *UIN*. At that point, the message type is changed to COORDINATOR and circulated once again, this time to inform everyone else who the coordinator is (the list member with the highest *UIN*) and who the members of the new ring are. When this message has circulated once, it is removed and coordinator starts to manage the ring. The coordinator is in charge of the following things: 1) Manage the token; 2) Receive service provider agent's advertisement; 3) Receive new application team request for proliferating a pair of new middle agents (host and its duplicate) for this new application team; 4) Receive a application team removing request for cancelling the related middle agents (host and its duplicate) and removing them from the ring; 5) Maintain the logical ring (control to proliferate new middle agents or remove old ones).

When the ring is initialized, the coordinator will produce a token. The token circulates around the ring. It is passed from middle agent k to k+1 (modulo the ring size, 256) in point-to-point messages. When a middle agent acquires the token from its neighbor, it has three aspects of tasks to do. Firstly, middle agent checks the token if there is a service provider agent that has just advertised it to the coordinator. If yes, the middle agent registers service provider agent's information. Secondly, if there is any maintaining message (new application team information, canceled an application team information, applied *UIN*, etc.) on the token, the middle agent updates related items or states. Thirdly, if the middle agent is a host, it checks if its

duplicate is normal (If not, it proliferates a duplicate and makes it on the ring). If the middle agent is a duplicate, it checks if its host is normal (If not, it acts as the host, proliferates a duplicate and makes it on the ring). When the coordinator holds the token, it will clear the token and reload it again according to the environment.

Now let us see the mechanism of matchmaking. Middle agents which organise themselves dynamically to satisfy the change of environment coordinate the interactions between application agent and service provider agent. The interactions are divided into three categories: service provider agent registration, new application-based group registration, and matching application agent with service provider agent. Figure 5.7 presents the interaction patterns.

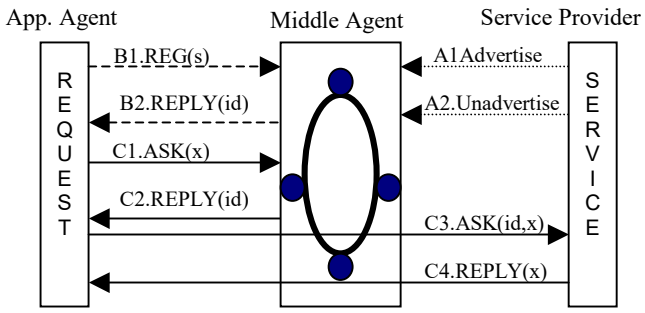

Figure 5.7 Interaction between agents

When a service provider agent wants to register for providing service to application agents (App. Agents), it advertises its capability and feature to the ***coordinator*** (A1). When the ***coordinator*** holds the token, it registers the service provider agent's information, puts the information on the token, and informs all other middle agents to register the agent's information as it did. When a service provider agent does not want to provide service to any application agent, it sends "unadvertise" to the ***coordinator*** (A2). The coordinator will remove its information and inform all other middle agents to remove the information as it did. If a middle agent detects a crashed service provider agent, it will ask ***coordinator*** to do the similar work as "unadvertise".

When a new application-based group wants to register for constructing its middle agents (host and duplicate), it sends a registration message to the ***coordinator*** (B1). When the ***coordinator*** holds the token, it will do the following things: 1) register

the new application-based group's information; 2) produce two middle agents for the application-based group and makes them on the ring; 3) put the information on the token; 4) inform all other middle agents to register the application-based group information as it did. When the token comes back, ***coordinator*** will clear this information from token and reply the new application-based group with related middle agent's access port number (B2). The application-based group informs all its application agents of the middle agent's port number. When an application-based group does not want to run forever, it sends "remove" to the ***coordinator***. The ***coordinator*** will remove the related middle agents from the ring and informs all other middle agents to remove the application-based group's information as it did.

For matching application agent with service provider agent, the steps are as follows:

1) Application agent asks its related middle agent to answer its request (C1).
2) Middle agent searches the ability and feature of each service provider agent, selects a service provider agent which is able to answer the request, and replies the agent's information to application agent (C2).
3) Since application agent knows which service provider agent is able to solve its request, it directly asks the service provider agent to answer the request (C3).
4) The service provider agent completes the request and replies the result to the application agent (C4).

After matching application agent with service provider agent, middle agent will not intervene in any interaction between application agent and service provider agent, that is, Steps 3 and 4 can be repeated for many times for other similar tasks.

5.4 Implementation of PAHIS

For verifying the performance of PAHIS, an application-based information-gathering system from WWW is developed. The middle agents (matchmakers) with ring-based architectural model match requester agents (application agents) with information retrieval agents (service provider agents). A prototype of the system for

information-gathering from WWW is implemented using C language and Socket technique in Unix environment.

The system consists of three parts, that is, application-based information-gathering agents, ring-based middle agents (A, B, C, and D. Assumed that D is the coordinator), and retrieval agents with their websites (M1 with W1, M2 with W2, …, Mx with Wx). The framework is shown in Figure 5.8. The relationships between information-gathering agents (Peer-to-Peer organisational structure) in each group are not described in Figure 5.8.

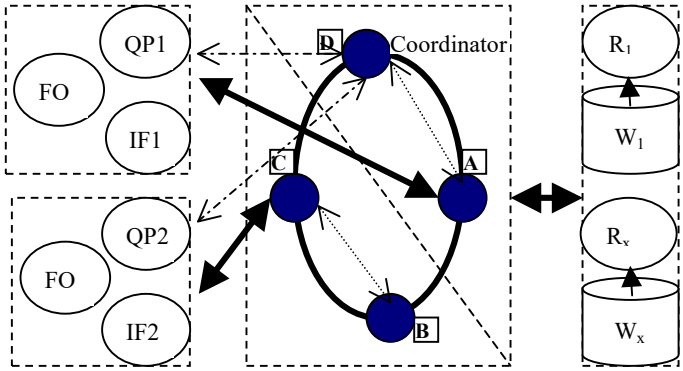

Figure 5.8 Framework for information gathering

We have designed two independent information-gathering groups (one for petroleum information gathering and another for financial investment information gathering). Each information-gathering group consists of Query Pre-processing agent (QPi), Information Filtering agent (IFi), and File Operating agent (FO). Here i = 1 or i = 2; it indicates group 1 or group 2. Because both groups operate the same file server, they share the only FO to operate documents (query statements, category-based training documents, and retrieved documents from websites) in file server (Li, Zhang et al. 2002b). The host middle agent A directly serves for group 1. The middle agent D is the duplicate of middle agent A and acts as the coordinator at the moment. The host middle agent C directly serves for group 2. The middle agent B is the duplicate of the middle agent C. The part of retrieval agents includes websites and retrieval agents (R1-Rx) each of which only directly operates a specific website.

QP interacts with users, for example, inputting query statement by keyboard, inputting category-based training documents by transmission, controlling to execute specialised function, or displaying results or state of the system. At the same time, it is in charge of the registration request and cancellation request of the group. Because query statement and category-based training documents are described in natural language, document pre-processing is another task of QP. Pre-processing task consists of two processes. The first one is to analyse training documents for generating the expressions used in IF training. The second one is to process user's query statement and rough documents that are the output of retrieval agents for generating the expressions used in information retrieval and filtering. The first task of IF is to train the expression weights based on category-based documents. The categories are defined according to users' information gathering purposes. A description, some relevant supporting documents, and some irrelevant supporting documents about one category are given for training the expression weights. The second task of IF is to filter the rough documents. The model of information filtering for each category consists of two stages, namely, expression weight training and rough document filtering. We applied the vector model based on clustering techniques to train and filter the expression weights (Li, Zhang et al. 2002b).

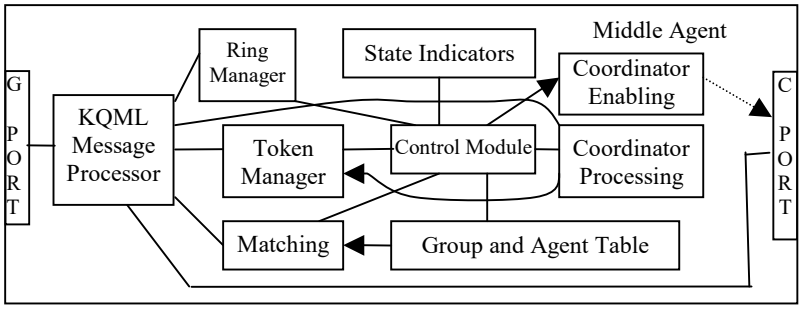

Figure 5.9 Structure of middle agent

The middle agent is of four aspects of tasks. Firstly, it matches application agent's request with a suitable service provider agent. Secondly, it may act as a coordinator (managing the application group and the service provider agent registrations and cancellations, etc.). Thirdly, it looks after its duplicate or vice

versa. Finally, it receives and transmits the token on the logical ring. The structure of middle agent is presented in Figure 5.9. A middle agent consists of nine components: Communication Protocol Module (G-PORT and C-PORT), KQML Message Processor (KMP), Coordinator Processing, Ring Manager, Token Manager, Group and Agent Table, State indicators, Matching, and Control Module.

The Communication Protocol Module deals with the interaction with other agents by means of KQML. In the lowest tier, there are two communication ports, one for general purpose (G-PORT) and another for coordinator purpose (C-PORT). All middle agents use the same port number to C-PORT for convenient registration of new application-based groups and service provider agents, but just one middle agent is available at any time. If a middle agent acts as a coordinator, its C-PORT is available; otherwise its C-PORT is prohibited. The Coordinator Enabling maintains this function. The KQML Message Processor (KMP) is in charge of the interpretation or generation of the KQML message. Once an incoming KQML message is detected by the G-PORT or C-PORT, it will be passed to the KMP. The KMP translates incoming KQML messages into a form that agents can understand, or vice versa. The Coordinator Processing deals with the coordinator election, generating a token when the ring is initialized, managing current systems, producing a pair of new middle agents for a new application-based group, removing a pair of middle agents for the cancelled application-based group, and managing the registration of the service provider agent. When the coordinator holds the token, it puts the registration information of the service provider agent in the token. Other middle agents will get the information from token and save it to the Group and Agent Table. If a service provider agent is cancelled, the related item will be removed from the Group and Agent Table. The Ring Manager deals with changing states of the middle agent (from duplicate to host), producing a duplicate and adding it on the ring. There are two indicators to indicate its successor and predecessor on the ring. The Token Manager deals with the receiving token, doing all tasks when a middle agent holds the token, updating Group and Agent Table and State indicators in accordance with the information on the token, and transmitting the token to its successor. The Group and Agent Table hold the information about service provider

agents and agent groups. The State indicators include middle agent state indicator, current system indicator, and the list of allocated middle agent numbers. The Matching Module matches application agent's request with a suitable service provider agent. The Control Module, which is the control centre of the middle agent, makes other components work in order.

We are interested in locating some information about "what information is available on petroleum exploration in the South Atlantic" that is stored on three websites, which contain more than 500 documents on various aspects of petroleum as an experiment. Certainly, the system has been trained by categorised petroleum information before the query. The result is that 307 documents are retrieved and 29 articles are got after filtering, but one out of 29 articles was non-relevant. The precision and recall rate of the filtering algorithm are about 97% and 9%, respectively. However, the results very depend on the training documents and the articles on the websites.

Scalability is an important issue in large systems. The systems should work properly even when more and more users and information resources joint the systems. We have investigated the scalability of self-organising architectural model associated with increased number of information gathering systems. Along with more and more information gathering systems are created, there will be more and more middle agents on the ring. In consequence token will need more time in a cycle. However, this does not affect matching performance because each information gathering system has its own matchmaker. The performance of ring-based matchmakers will change with creating or cancelling an information gathering system. The performance of the matchmakers will abate to the lowest point while the coordinator is elected and the ring is initialized.

We have measured the interval from the ring establishment to the registrations of all application-based groups and retrieval agents. The following are the results: the establishment of the ring takes about 67% of the time interval; the registrations of the teams take about 29% of the time interval; the registrations of the service provider agents take about 4% of the time interval. The cancellations of application group and service provider agent take the similar time with their registrations. From

the above results, it takes more time that the ring is established because of the coordinator election. However, the registration and cancellation of the service provider agent are very fast.

5.5 Evaluation of PAHIS' Structure

We have evaluated the ring-based architectural model with performance predictability, adaptability, and availability. The performance predictability can be measured by complexity and efficiency of the system. Adaptability is measured by extendibility of the system (Li, Zhang et al. 2003a). The result of this evaluation has shown that the ring-based middle agent architectural model is competent in agent-based systems with middle agents.

5.5.1 Complexity

Complexity measures the number of links among agents or middle agents in organisational structure. The more the number of interactions between agents is, the more complex agent-based system is. If agent-based system is designed by using the peer-to-peer structure, the complexity of the system will be highest because agents in peer-to-peer structure are fully connected. Complexity (C) is defined as following.

$$C = Number\ of\ links\ among\ agents$$

For each organisational structure, the detail formula for the complexity is different. The formulae for each structure are defined as following (C_p means the complexity of peer-to-peer structure; C_f means the complexity of grouping with facilitator structure; C_t means the complexity of tree structure; C_r means the complexity of ring structure).

$$C_p = n(n-1)/2$$

where n is the number of agents.

$$C_f = f(f-1)/2+\sum(n_i(n_i+1)/2)$$

where f is the number of middle agents; n_i is the number of agents under the i^{th} middle agent.

$$C_t = sum(in\text{-}degree(f_i), out\text{-}degree(f_i))$$

where f_i means i^{th} middle agent ($i = 1,2...n$); n is the number of middle agents in the tree structure.

$$C_r = N_f+N_s+2m+mN_s+\sum(n_i(n_i-1)/2)$$

where N_f means the number of middle agents; N_s means the number of service provider agents; n_i means the number of requester agents in Group i ($i = 1,2...m$; m is the number of groups).

According to the prototype described in Figure 5.8 (three service providers), the C_r is 26, which is worse than Tree (less than 20) but better than Peer-to-Peer and Grouping with facilitator (between 30 to 70).

5.5.2 Efficiency

Efficiency measures the number of links from service request to completion of the service. Efficiency (E) is defined as following.

$$E = Number\ of\ interactions\ until\ completion\ after\ receiving\ request$$

The formulae for each structure are defined as following (E_p means the efficiency of peer-to-peer with contract-net coordination mechanism; E_f means the efficiency of grouping with facilitator; E_t means the efficiency of tree structure with broker coordination mechanism; E_r means the efficiency of ring structure).

$$E_p = (n-1)*3+1$$

where n is the number of agents.

$$E_f = N_l+N_f+N_r$$

where N_l is the number of interactions among agents under a local middle agent; N_f is the number of interactions among local middle agents; N_r is the number of interactions among agents under the remote middle agent.

$$E_t = N_l+N_f+N_r$$

where N_l is the number of interactions between agents and local middle agent; N_f is the number of interactions between local middle agent and global agent; N_r is the number of interactions between remote middle agent and agents under it.

$$E_r = N_l+N_f+N_r+N_s$$

where N_l is the number of interactions among agents under a group; N_f is the number of interactions between agent and its middle agent; N_r is the number of interactions among middle agents; N_s is the number of interactions between requester agent and service provider agent.

According to the prototype described in Figure 5.8, the E_r is 20, which is better than Tree, Peer-to-Peer and Grouping with facilitator (Between 30 to 50).

5.5.3 Extendibility

Extendibility is to evaluate the adaptability of agent-based system. When a new type of service request is introduced into the system, it is necessary to reconfigure the existing system. Reconfiguration includes change, replacement, deletion and addition of agent or middle agent. Extendibility (*EL*) measures the resources that need to add an agent or middle agent to the existing system. It is defined as following.

$$EL = Number\ of\ links\ that\ need\ to\ add\ agent\ or\ middle\ agent$$

The formulae for each structure are defined as following (EL_p means the extendibility of peer-to-peer structure; EL_f means the extendibility of grouping with facilitator structure; EL_t means the extendibility of tree structure; EL_r means the extendibility of ring structure).

$$EL_p = n$$

where n is the number of agents.

$$EL_f = n+1\ (if\ an\ agent\ is\ added)\ or$$

$$EL_f = 2\ (if\ a\ middle\ agent\ is\ added)$$

where n is the number of agents under the middle agent.

$$ELt = 1$$

ELr = n (if a requester agent is added) or

ELr = 4 (if a pair of middle agents are added) or

ELr = 1 (if a service provider agent is added)

where *n* means the number of requester agents in the group.

According to the prototype described in Figure 5.8, the maximum of the EL_r is 4, which is worse than Tree (1) but better than Peer-to-Peer and Grouping with facilitator (7, 12 respectively). From this measure, we can reason whether an organisational structure can be easily extend. Ring structure is more scalable than peer-to-peer and grouping with facilitator.

5.5.4 Availability

When agent or middle agent is abnormal, the behaviours of the agent-based system may be erroneous. The abnormal situations can be begot by fault or malicious action. If local middle agent in tree structure is abnormal, agents under the middle agent cannot act and consequently the availability of system decreases. Availability (*A*) is defined as following.

$$A = (C-F)/C$$

where: *C* means Complexity;

F = Number of links connected to abnormal agent or middle agent

The formulae for each structure are defined as following (A_p means the availability of peer-to-peer structure; A_f means the availability of grouping with facilitator; At means the availability of tree structure; A_r means the availability of ring structure).

$$A_p = (C_p–F_p)/C_p$$

$$F_p = n\text{-}1$$

where *n* is the number of agents.

$$A_f = (C_f–F_f)/C_f$$

F_f = sum(the number of peer middle agents, the number of agents connected to the fault middle agent)

$A_t = (C_t – F_t)/C_t$

$F_t = 1$ *(If an agent is abnormal) or*

$F_t = sum(in\text{-}degree(f_i), out\text{-}degree(f_i))$ *(If a local middle agent is abnormal) or*

$F_t = C_t$ *(If global middle agent is abnormal)*

where f_i means i^{th} middle agent ($i = 1,2…n$); n is the number of middle agents in the tree structure; out-degree(f_i) includes all offspring links of f_i.

$A_r = (C_r – F_r)/C_r$

$F_r = n_i$ *(If a requester agent is abnormal) or*

$F_r = 4$ *(If a middle agent is abnormal) or*

$F_r = 1$ *(If service provider is abnormal)*

where n_i means the number of requester agents in Group i ($i = 1,2…m$; m is the number of groups).

According to the prototype described in Figure 5.8, the minimum of the A_r is 0.85, which is similar with Tree, Peer-to-Peer and Grouping with facilitator (between 0.8 to 0.9). If the coordinator in ring structure is abnormal, A_r will decrease to a lower point (about 0.65). However, the coordinator can be automatically regenerated if it is abnormal.

5.6 Summary

For developing PAHIS, an application-based information-gathering system from WWW is developed. There are three categories of agents in the system, namely, application agent category, middle agent category, and service provider category. A self-organising ring-based architectural model is proposed to organise the middle agents in PAHIS. The ring-based architectural model is based on logical ring organisational structure and token ring coordination mechanism. About the organisational structure, the middle agents are divided into hosts and duplicates according to their roles in PAHIS. About the coordination mechanism, coordinator

electing mechanism is employed to build the logical ring, manage token, and produce or cancel middle agents based on agent group concept.

A prototype of the system for information-gathering from WWW has been implemented using C and Socket technique in Unix environment. The results of the system have shown that PAHIS with ring-based architectural model can be used as a foundation of agent-based HIS. Furthermore, it is undoubted that PAHIS based on the ring architectural model is more powerful in predictability, adaptability, and availability by evaluation of the organizational structures adopted by agent-based systems. The result of the evaluation has shown that the ring-based architectural model is competent in agent-based systems with middle agents.

Chapter 6

6 Case Study 2: Financial Investment Planning System

For verifying the HIS construction ability of MAHIS, we have developed an agent-based financial investment planning system. This system gives advice to the investors. Diverse models (techniques) are integrated in the agent-based financial investment planning system. The techniques include: financial risk tolerance mode based on fuzzy logic, asset allocation model based on fuzzy logic, portfolio selection models (Markowitz's model, fuzzy probability portfolio selection model, and possibility portfolio selection mode), interest prediction models (feed forward network model, and combination of fuzzy logic and genetic algorithms model), and ordered weighted averaging (OWA) operators for result aggregation. All these models were discussed by Zhang and Zhang (Zhang and Zhang 2004). In our financial investment planning system, the aforementioned models are directly employed. The distinct feature of our financial investment planning system is developed by following MAHIS. Section 6.1 presents the conceptualisation phase including financial investment planning requirements and hybrid strategy identification. Section 6.2 discusses the analysis phase. The system is analysed by

means of six models. Section 6.3 is the design phase. The system and agent structures are described. The system is implemented based on PAHIS (Section 6.4).

6.1 Conceptualisation

In financial investment planning, a large number of components that interact in varying and complex ways are involved. This leads to complex behaviour that is difficult to understand, predict and manage (Zhang and Zhang 2004). Take one sub-task of financial planning – financial portfolio management – as an example. The requirements of the financial portfolio management are given in this section. After that, the intelligent techniques, hybrid technique relations, and hybrid strategy adopted by the system are mentioned.

6.1.1 Financial Investment Planning Requirements

The task environment has many interesting features (Zhang and Zhang 2004), including:

- the enormous amount of continually changing, and generally unorganised information available;
- the variety of kinds of information that can, and should, be brought to bear on the task (market data, financial report data, technical models, analysts' report, breaking news, etc.); and
- many sources of uncertainty and dynamic change in the environment. It is obvious that financial planning is typically a complex problem for which hybrid solution is crucial.

Besides the features of the environment of the financial investment planning, we would like to give a typical scenario for investment to indicate the concrete requirements. In order to identify which components should be contained in a typical financial investment planning system, without loss of generality, consider a financial establishment house providing investment advice for clients. In such a

place, there are: an up-front administrator, one or more personnel officer(s), and many financial investment experts (decision makers). The advice giving (decision making) process is initiated by an investor contacting the up-front administrator with a set of requirements. The administrator asks the personnel officer to provider the experts' profiles, and then delegates the task to one or more experts based on the 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 administrator sends the advice to the investor.

When an investor wants to invest some money, he/she usually goes to a financial investment adviser for advice. The first thing the adviser needs to do is to understand the investor's individual circumstances (*IIC*). The adviser may ask the client to provide the following information about him/her: his/her financial position (for example, annual income and total net-worth), age, tax effectiveness, etc. Based on the information, the adviser will evaluate the financial risk tolerance ability (*FRT*) as well as the investor's investment goals (*IIG*).

If the investor's primary goal is income, then investments that provide interest or dividend payments regularly and dependably are required. If the primary goal is growth, then investments that are likely to increase in value are appropriate so that they may be resold for more than their initial cost. If, however, the primary goal is to avoid risk, then investments that offer the greatest safety of principal, and protection from inflation are required (Zhang and Zhang 2004).

When giving investment advice to an investor, the first thing a financial investment planning system needs to do is to determine the investor's investment policy (*IIP*). Based on this (aggressive or conservative etc.), the system can then decide in which categories (stock market, real estate, etc.) the investor should invest. Suppose the adviser, after evaluating the investor's financial risk tolerance ability, suggests that the investor invests in the stock market. How can the advisor select a portfolio for the investor taking into account the individual constraints of that

particular investor (e.g. risk tolerance level and return rate)? The adviser must first gather some information as market data, financial report data, technical models, analysts' reports, breaking news. After gathering the information, the adviser then makes a portfolio selection decision based on certain models (the Markowitz, the fuzzy probability, and the possibility portfolio selection).

According to the aforementioned requirement descriptions, three use cases, namely, *ask investment advice*, *ask profiles*, and *delegate requirement*, are schemed out. The *ask investment advice* use case is presented in Figure 4.2 and Figure 4.3. Other four figures are omitted for shortening the length of this thesis.

6.1.2 Hybrid Strategy Identification Model

From the financial investment planning requirements, we have known that eight processes need intelligent techniques. They are: financial risk tolerance evaluation (*RTE*), asset allocation (*AA*), three portfolio selections (*PS1, PS2, and PS3*), two interest predictions (*IP1* and *IP2*), and ordered weighted averaging (*OWA*). The intelligent technique characteristics of those processes are summarised in Table 6.1. The intelligent techniques, namely, fuzzy logic, neural networks, and combination of fuzzy logic and genetic algorithm (FL and GA), are selected.

Table 6.1 The intelligent characteristics of the processes

Process	Characteristics	Intelligent technique
RTE	Fussy set; if-then rules	Fuzzy Logical
AA	Fussy set; if-then rules	Fuzzy Logical
PS1	Matrix and calculating; quadratic programming	Markowitz Model
PS2	Fuzzy set; probability; similarity degree	Fuzzy Logic
PS3	Possibility distribution; vector;	Possibility Model
IP1	Uncertain prediction; matrix data	Neural Networks
IP2	Fuzzy set; evolutionary computing	FL and GA
OWA	if-then rules; inference; fuzzy set	Expert System

From the processes of the financial investment planning and the environment of the system, two hybrid integration strategies (*MHS* and *AHS*) should be selected.

The *MHS* is *tight-coupling* and the *AHS* should be *fully-integration*. This is because messages exchanged between intelligent processes and other components are small and they may be directly transmitted by agent communication language rather than by data files.

Because the *MHS* is *tight-coupling*, the ⫦ relation is selected as the hybrid technique relation by following the relation selection mechanism described in Section 4.3.2. So the *tight-coupling* and ⫦ are selected as the hybrid strategy and hybrid technique relation, respectively.

6.2 Analysis of the System

Based on the description in Section 6.1, the process organizational relationships, tasks, agents, groups, categories, VOs, system dynamics, and interactions between agents, groups, and categories are discussed in this section. The analysis is divided into six stages in accordance with the analysis models of MAHIS.

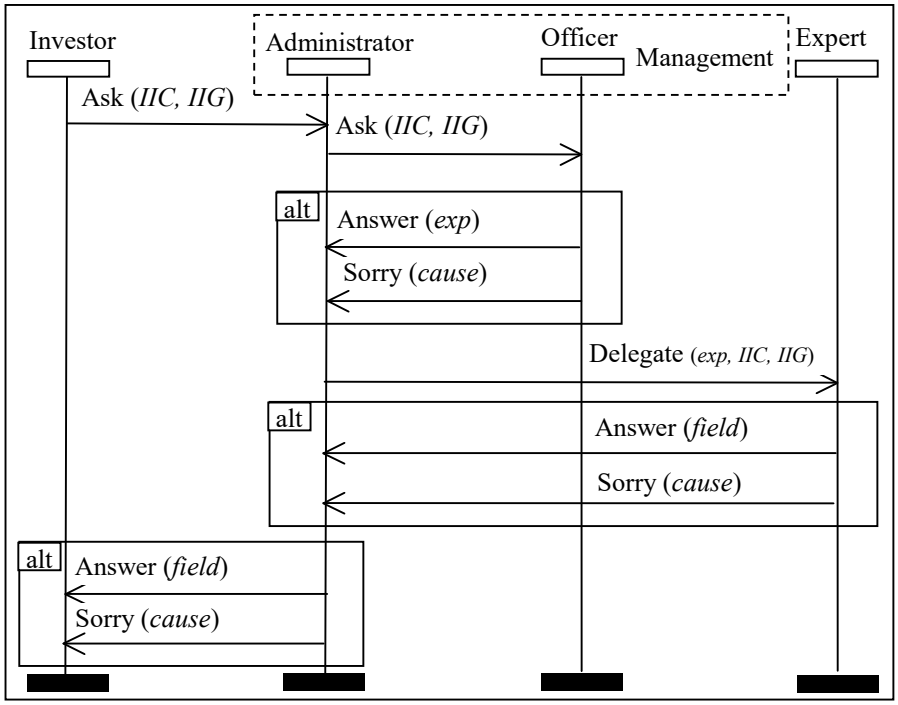

Figure 6.1 MSC process of financial investment planning

6.2.1 Organisation model

The first task of the organisation model is to obtain the problems and opportunities in the wider organisational context as described in Table 4.2. This task is obviously based on the requirements. To avoid redundancy, the problems and opportunities of the financial investment planning are omitted here. In the second part of the organisation model, we can divide the financial investment planning system into three organisations: investors, management, and experts. Figure 4.6 has shown the organisations and their relationships. Figure 6.1 presents the interactions between the people in each organisation.

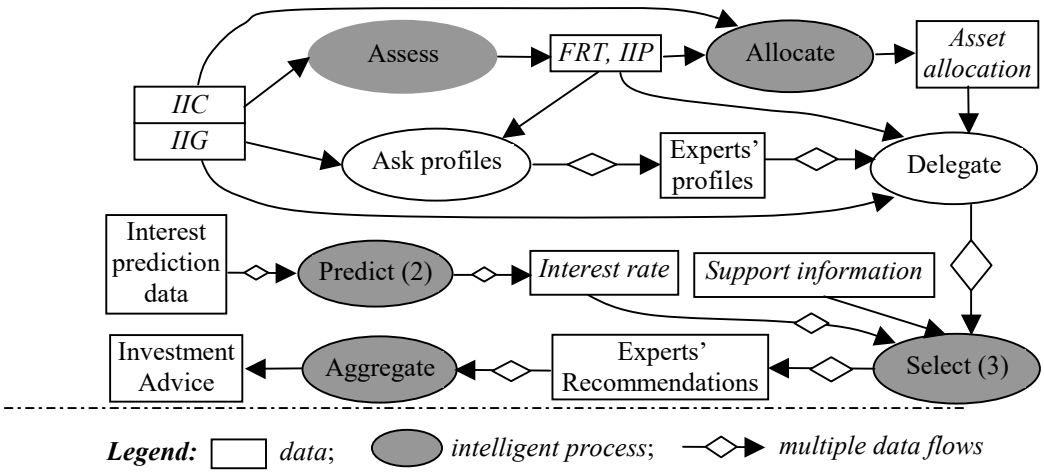

Figure 6.2 Relationships between processes and resources

The investor organisation includes two processes (investor information collecting and investment advice presenting), two kinds of people (investors and administrator), and some knowledge for making plans. The management organisation includes four processes (task planning, expert profile maintaining, expert matchmaking, and request delegating), two kinds of people (administrator and personnel officers), and some knowledge for selecting experts. The expert organisation includes eight processes (financial risk tolerance assessing, asset allocating, Markowitz portfolio selecting, fuzzy probability portfolio selecting,

possibility portfolio selecting, feed forward network interest predicting, combination interest predicting, and portfolio results aggregating), one kind of people (experts), and some knowledge for intelligent processes. Figure 6.2 presents the relationships between processes and resources.

Because tight-coupling is selected as the financial investment planning hybrid strategy, it is wise to categorise the processes in accordance with the organisations. Furthermore, the processes should be grouped according to the interactions among them.

6.2.2 Task Model

The *overall task of investment advice* can be decomposed into two subtasks: *interaction with user* and *investment planning*. The *interaction with user* can be further decomposed into *investor information collection* and *advice presentation*. The *investment planning* can be decomposed into three subtasks: *work planning*, *expert management*, and *request delegation*. The *expert management* can be decomposed into *expert profile maintenance* and *expert matchmaking*. The *request delegation* can be decomposed into *financial risk tolerance assessment*, *asset allocation*, *portfolio selection*, *interest prediction*, and *result aggregation*. Figure 4.7 presents some tasks in task tree. Because the tasks are in accordance with the processes, the relationships of these tasks are similar with the ones of the processes as shown in Figure 6.2.

6.2.3 Agent Model

The agents can be identified based on the processes and tasks according to the agent identification strategies described in Section 4.4.3.

The *information collection* and *advice presentation* tasks can be carried out by an agent: *Interface Agent*. The *expert profile maintenance* and *expert matchmaking* tasks can be carried out by an agent: *Middle Agent*. The *work planning* task can be

carried out by *Planning Agent*. The *financial risk tolerance assessment* task is carried out by *Financial Risk Tolerance Assessment Agent (FRTA Agent)*. The *asset allocation* task is carried out by *Asset Allocation Agent (AA)*. The *portfolio selection* task is carried out by *Markowitz Portfolio Selection Agent*, *Fuzzy Probability Portfolio Selection Agent*, and *Possibility Portfolio Selection Agent*. The *interest prediction* task is carried out by *Feed Forward Network Interest Prediction Agent* and *Combination Interest Prediction Agent*. The *result aggregation* task is carried out by *Decision Aggregation Agent*. Figure 4.8 shows the planning processing behaviour for planning agent in *activity diagram*. Figure 4.9 shows it in *state chart*. Here, we give the interface agent in activity diagram as shown in Figure 6.3. Other agents' graphic representations are omitted for saving space.

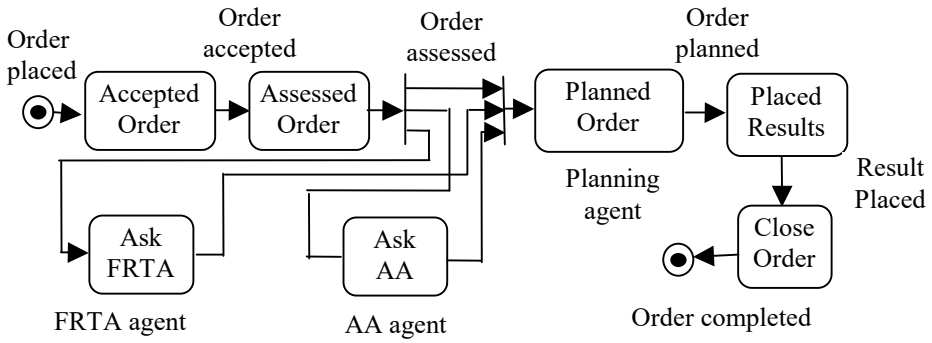

Figure 6.3 Activity diagram for interface agent

6.2.4 Coordination Model

According to the scenarios between the organisations described in Figure 6.1, the prototypical scenarios between agents are easily schemed out. These kinds of MSC are omitted for saving space. The events of the planning agents have presented in Figure 4.11. Now we present the events of user handling agents in Figure 6.4.

From Figure 4.11 and Figure 6.4, the interactions and events can be obtained. The interactions include *ask*, *tell*, and *sorry*. The data include *IIC*, *IIG*, *expert name*, *invest field*, *invested field*, *agent name*, and *context*. The sequence of the interactions has indicated in the event flow diagrams with digitals (subscript digital

behind each interact). The interaction with smaller number will more previously occur than the one with bigger number. Some of the interactions have been described in Figure 4.12 with SDL state diagrams.

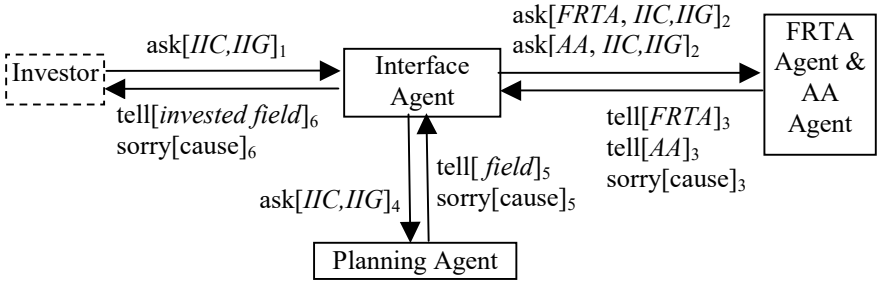

Figure 6.4 The events of user handling agents

6.2.5 Expertise Model

To make decisions, the agents must accomplish some reasoning based on their knowledge, and on other available information. The expertise model indicates the different levels of knowledge that agents should have. In the financial investment planning system, the agents should have three levels of knowledge. The first level of knowledge is for agent interaction and communication. This involves, for example, domain-specific and domain-independent terminologies and their relationships. The identified domain-dependent terms and their relationships will result in the construction of a domain-specific ontology for a specific application. The second level of knowledge is some domain knowledge related to specific problem solving techniques. The third level of knowledge is meta knowledge that directs the activities of an agent. For example, the following fuzzy rules are the knowledge about financial risk tolerance evaluation agent:

Rule 1: If the investor's annual income (*AI*) is low (*L*), and the investor's total net worth (*TN*) is low (*L*), then the investor's risk tolerance (*RT*) is low (*L*);

Rule 2: If *AI* is *L* and *TN* is *M* (medium), then *RT* is *L*;

Rule 3: If *AI* is *L* and *TN* is *H* (high), then *RT* is *MO* (moderate);

Rule 4: If *AI* is *M* and *TN* is *L*, then *RT* is *L*;

Rule 5: If *AI* is *M* and *TN* is *M*, then *RT* is *M*O;

Rule 6: If *AI* is *M* and *TN* is *H*, then *RT* is *H*;

Rule 7: If *AI* is *H* and *TN* is *L*, then *RT* is *MO*;

Rule 8: If *AI* is *H* and *TN* is *M*, then *RT* is *H*;

Rule 9: If *AI* is *H* and *TN* is *H*, then *RT* is *H*.

These rules need to be represented in CML as following:

```
KNOWLEDGE-MODEL financial-investment-planning;
DOMAIN-KNOWLEDGE investor-domain;
DOMAIN_SCHEMA financial-investment-schema;
CONCEPT investor;
DESCRIPTION:
    "A description of an investor in the database of financial investment planning system";
ATTRIBUTES:
    name: STRING;
    age: NATURAL;
    annual-income: REAL;
    total-net-worth: REAL;
    invest-amount: REAL;
    invest-attitude: {aggressive, conservative};
    aggressive-level: NATURAL;
    ai: ai-value;
    tn: tn-value;
    rt: rt-value;
AXIOMS:
    0<=aggressive-level <= 10;
END CONCEPT investor;
END DOMAIN-SCHEMA financial-investment-schema;
VALUE-TYPE ai-value;
    TYPE: ORDINAL;
    VALUE-LIST: {"L","M","H"};
END-VALUE-TYPE ai-value;
VALUE-TYPE tn-value;
    TYPE: ORDINAL;
```

```
    VALUE-LIST: {"L","M","H"};
END-VALUE-TYPE tn-value;
VALUE-TYPE rt-value;
    TYPE: ORDINAL;
    VALUE-LIST: {"L","MO","H"};
END-VALUE-TYPE rt-value;
KNOWLEDGE-BASE financial-risk-evaluation;
USES:
    investor FROM financial-investment-schema;
EXPRESSION:
    investor.ai = "L" AND investor.tn = "M"
    INDICATES
    investor.rt = "L";
    investor.ai = "L" AND investor.tn = "H"
    INDICATES
    investor.rt = "MO";
    investor.ai = "M" AND investor.tn = "L"
    INDICATES
    investor.rt = "L";
    investor.ai = "M" AND investor.tn = "M"
    INDICATES
    investor.rt = "MO";
    investor.ai = "M" AND investor.tn = "H"
    INDICATES
    investor.rt = "H";
    investor.ai = "H" AND investor.tn = "L"
    INDICATES
    investor.rt = "MO";
    investor.ai = "H" AND investor.tn = "M"
    INDICATES
    investor.rt = "H";
    investor.ai = "H" AND investor.tn = "H"
    INDICATES
```

```
    investor.rt = "H";
END KNOWLEDGE-BASE financial-risk-evaluation;
END KNOWLEDGE-MODEL financial-investment-planning;
```

6.2.6 Reorganisation Model

The financial investment planning system includes three organisations: investors, management, and experts as described in Section 6.2.1. We can divide all the agents into three categories: *application agent*, *middle agent*, and *intelligent processing agent* in correspondence with the three organisations. These three categories of agents correspond to the application agent, middle agent, and service provider of PAHIS. The three agent categories can be organised with hierarchy structure as shown in Figure 6.5.

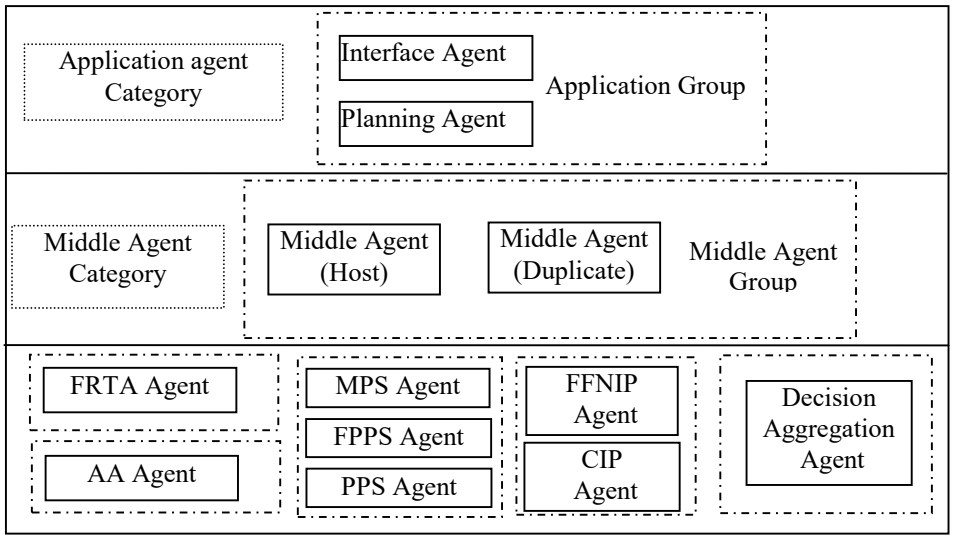

Figure 6.5 Agents, groups and categories

The application agent category includes *Interface Agent* and *Planning Agent*. *Interface Agent* and *Planning Agent* compose an application group. The application

group is the primitive member of the application agent category. The primitive member of the application category can dynamically change according to the user request.

The middle agent category includes the some tasks of administrator and all tasks of personnel officers. Each group in middle agent category includes two middle agents and serves for one specific group located in the application agent category. The number of groups in the application agent category is the same with the number of groups in the middle agent category. If an application group is added into the system, a new group in the middle agent category will be automatically produced.

The intelligent service provider agent category includes five groups: *financial risk tolerance assessment*, *asset allocation*, *portfolio selection*, *interest prediction*, and *result aggregation*. Each group includes one or more intelligent processing agents. The *financial risk tolerance assessment* group includes *Financial Risk Tolerance Assessment (FRTA) Agent*. The *asset allocation* group includes *Asset Allocation (AA) Agent*. The *portfolio selection* group includes *Markowitz Portfolio Selection (MPS) Agent*, *Fuzzy Probability Portfolio Selection (FPPS) Agent*, and *Possibility Portfolio Selection (PPS) Agent*. The *interest prediction* group includes *Feed Forward Network Interest Prediction (FFNIP) Agent* and *Combination Interest Prediction (CIP) Agent*. The *result aggregation* group includes *Decision Aggregation Agent*. New developed intelligent processing agents can be added into the system by means of advertising themselves to the category and indicating which group the agents prefer. That is, the group in this category is managed by middle agent rather than the intelligent processing agent category.

The VO will be automatically defined when an application is run. The member of a VO includes all agents in the application group and relevant intelligent processing agents which are called by the running application.

Because the financial investment planning system has the analogical category role, group role, and VO role adopted by PAHIS, PAHIS can be used in this system as the infrastructure. See Section 5.2 about those roles and dynamics rules.

6.3 Design of the System

Having completed the analysis of the system, the design phase follows. The first model to be generated is the global architectural model (Figure 6.6). This shows that the financial investment planning system is based on PAHIS. Because there is just one application group in the system, only two middle agents (host and its duplicate) are generated in PAHIS in correspondence with the application group. Certainly, PAHIS supports more than one application groups which can be added into the platform dynamically.

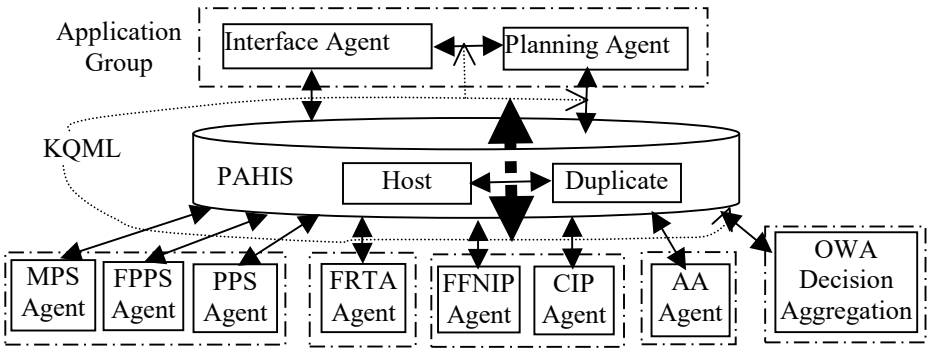

Figure 6.6 Architecture of financial investment planning system

When the platform is initiated, all intelligent processing agents and the application group must register themselves to the platform first. The planning agent in the application group has its own domain-specific knowledge base as well as meta-knowledge for using the intelligent processing agents. The middle agent records the capabilities, ontology, names, and group of all the intelligent processing agents. The scenario goes as follows.

At a certain stage of the financial investment planning process, the planning agent sends a KQML message using the *recommend-one* performative to its middle agent, according to its meta-knowledge. The middle agent then retrieves its intelligent technique agent database and replies with an appropriate intelligent technique agent's name and ontology, which has the capability asked for using the *reply* performative. After that, the planning agent communicates directly with the

intelligent technique agent to solve a specific problem. The planning agent provides the intelligent processing agent with some parameters according to the ontology, and the intelligent processing agent sends the results to the planning agent. When an intelligent agent needs help from other intelligent technique agent, it sends KQML message to the coordinator, which is in charge of the registrations and cancellations of the intelligent technique agents and application groups in PAHIS platform.

Based on the above description, the internal structures of the agents in the system can be identified (see Figure 6.7). All agents have a KQML message processor (KMP) and an ontology interpreter. The KMP translates incoming KQML messages into a form that agents can understand, or vice versa. The agents need the ontology interpreter to decrypt and processing the "*:content*" part of the KQML massage when they solve a problem.

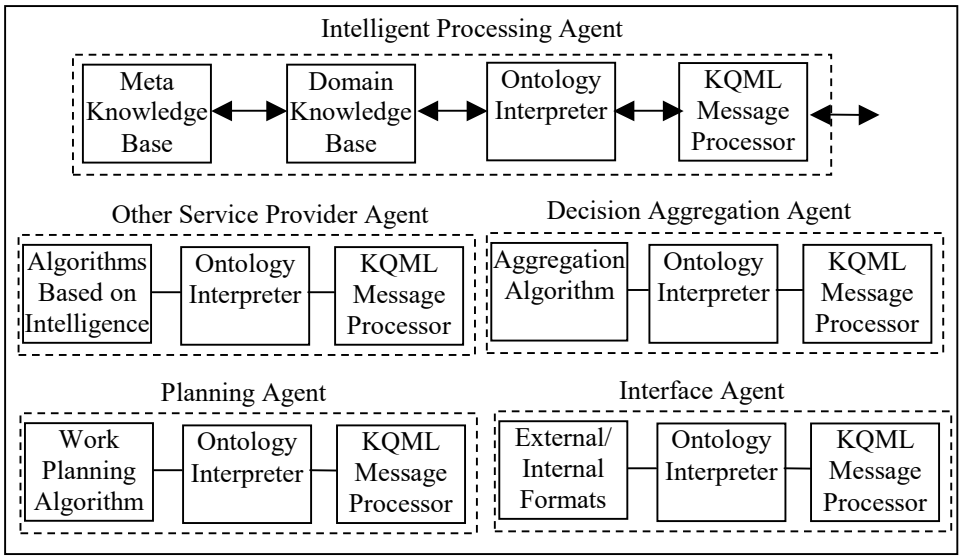

Figure 6.7 Structure of the agents in the system

The domain knowledge in intelligent processing agents is not usually adequate to make a decision. Relevant information and skills to use the knowledge and information are also needed. The help of service provider agents for data pre- and/or post-processing is often required. The meta-knowledge of decision making agents advises them when help is required from service provider agents.

6.4 Implementation of the System

Under the support of PAHIS, the prototype of the financial investment planning system is implemented. The system has employed the implementation of the middle agent in PAHIS described in Section 5.4. However, some modules of the middle agent are redeveloped. The *Matching module* which carries out the expert profile maintenance task and expert matching task is a part of the *Group and Agent Table* in middle agent. Some ideas have been borrowed from "the financial investment planning system" described in (Zhang and Zhang 2004) to propose and develop all other agents.

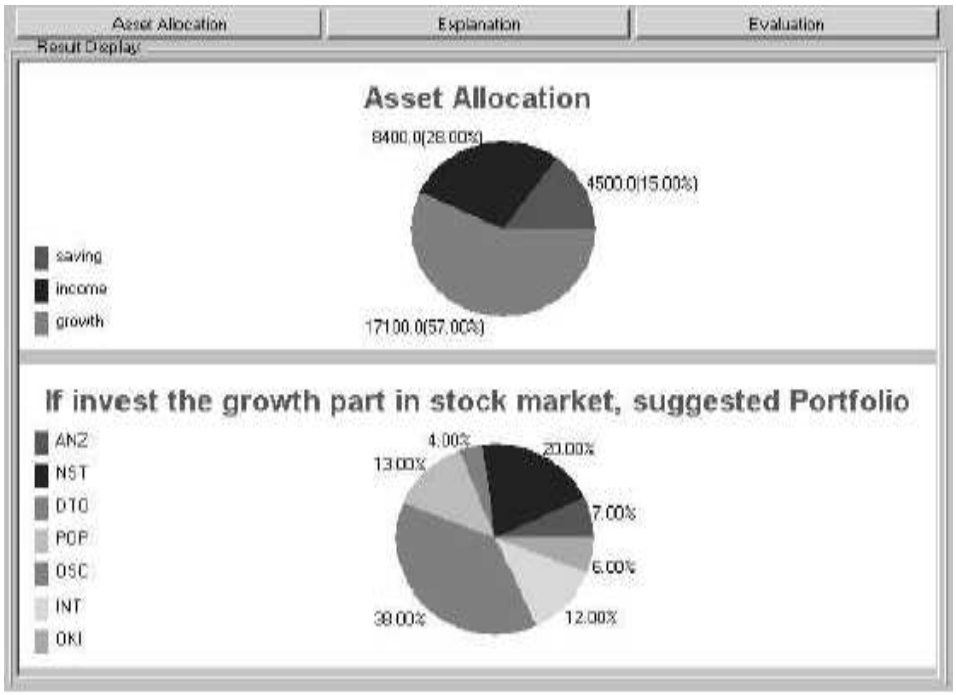

Figure 6.8 Example of asset allocation and portfolio results

The system can provide reasonable financial investment planning information based on data provided by user, and some relevant models. Figure 6.8 shows asset allocation results when the annual income is $50,000, net worth $800,000, age 35, investments $30,000, and investment attitude is aggressive. By clicking the "explanation" button, the corresponding explanation of how to get the results is

displayed in the "result display" window. If the growth section is invested in the stock market, the system can provide a portfolio for the user. The portfolio is the aggregated result of three portfolio based on the Markowitz portfolio selection model, the fuzzy probability portfolio selection model, and the possibility distribution portfolio selection model. By clicking the "evaluation" button, the system will provide the comparisons of the produced portfolios.

6.5 Summary

The financial investment planning system developed by following MAHIS has the following crucial characteristics that differentiate this research from others: (1) The planning agent in the application group can easily access all the intelligent processing agents, including financial risk tolerance assessment agent, asset allocation agent, portfolio selection agents, interest prediction agents, and decision aggregation agent, available in the system whenever needed. At the same time, one intelligent processing agent can ask other intelligent processing agents for help; (2) The intelligent processing agents can be added to or removed from the system dynamically; (3) The presence of PAHIS in this system supports the VO concept and makes the agents in the application category and intelligent processing category reusable; (4) Overall system robustness is facilitated through the use of the redundant middle agents with ring-based architectural model.

Chapter 7

7 Case Study 3: Petroleum Reservoir Characterisation

A petroleum reservoir is a volume of porous sedimentary rock which has been filled with hydrocarbon, such as oil or gas. Petroleum reservoir characterisation is a new practical technology for synthetically studying and evaluating petroleum reservoir. Since the complexity of the underground geology, a single intelligent technique can not solve the complicated and elaborate geologic problems. It is necessary that those geologic problems are synthetically studied by combining the multiple intelligent techniques (Soleng 1999).

The petroleum reservoir characterisation system starts with the digitisation of well logs parameter graphs based on expert system. Identification of lithology and prediction of porosity and permeability can be conducted by using hybrid intelligent techniques. The system has integrated four intelligent technique agents (complicated lithology identification with parthenogenetic algorithm, porosity prediction with neural network, permeability estimation with fuzzy neural network, and well logs curve-digitizing with expert system) based on PAHIS. We develop the petroleum reservoir characterisation system with the process stages of MAHIS. Section 7.1 presents the conceptualisation phase including petroleum reservoir

characterisation requirements and hybrid strategy identification. Section 7.2 discusses the analysis phase. The system is analysed by means of six models. Section 7.3 is the design phase. The system and agent structures are described. The system is implemented based on PAHIS (Section 7.4).

7.1 Conceptualisation

The petroleum reservoir characterisation includes many research subjects, e.g., prediction and calculation of parameters of reservoir properties, distribution of pressure, lithofacies identification. However, lithology, permeability, and porosity are the three most important parameters. The requirements of the identification of lithology and the prediction of permeability and porosity are given in this section. After that, the intelligent techniques, hybrid technique relations, and hybrid strategy adopted by the system are discussed.

7.1.1 Reservoir Characterisation Requirements

Reservoir properties are a set of parameters which are usually used to recognise the geologic information in spatial variability. Lithology, permeability, and porosity are widely used to determine the oil well or field production rate of hydrocarbon. Well logs are a series of multi-type digital measurements along the vertical depth of drilled wells (Huang, Wong et al. 1996). Understanding the form and spatial distribution of these heterogeneities is fundamental to the successful characterisation of petroleum reservoirs (Wong, Gedeon et al. 1995). For identification of lithology and prediction of porosity and permeability, rock samples are obtained by using a coring barrel to recover intact cylindrical samples of reservoir rock. These samples are then sent to the laboratory and different petrophysical properties (porosity, permeability, etc.) and lithofacies are measured (Wong, Bruce et al. 2002). Well log readings (simply called well logs) are obtained every 150mm or so of depth, by lowering various sondes in the drilled wells. These

measure formation and fluid properties in and around the wellbore location. Typical sondes generate electrical signals from measurements of radioactive, resistivity, acoustic, and neutron attenuation and scattering properties of the formation and its contained fluids. Because coring is a relatively time consuming and expensive process, much effort is made to relate other measures to the available core lithology, porosity, and permeability measurements so that the transformations developed can be applied to predict lithology, porosity, and permeability data in uncored wells or intervals (Huang, Wong et al. 1996).

However, a large amount of well logs, which were produced previously, faces to satisfy the current computation environment. Before computers were widely applied in petroleum industry, the well logs might be drawn on parameter graphs in curve mode (Li, Zhang et al. 2003c, Li, Zhang et al. 2003d). In current computation environment, a novel system with the ability to digitise and manage those legacy well logs is required in petroleum industry.

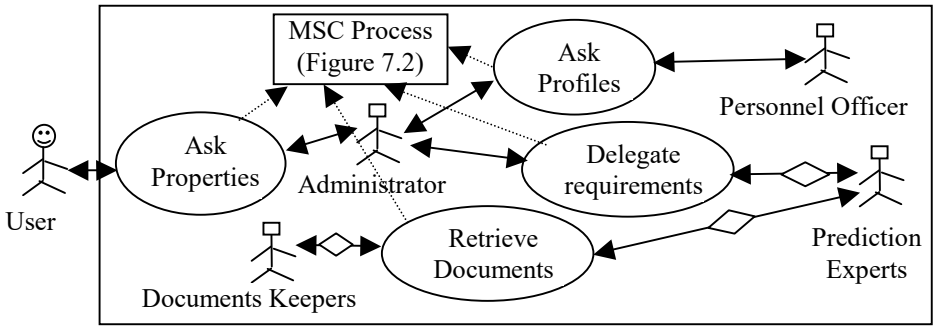

Figure 7.1 Use case of reservoir property prediction

In petroleum reservoir characterisation system, a large number of shared data and components that interact in variational, complex ways are involved. This leads to complex behaviour that is difficult to understand, predict and manage. Take one sub-task of petroleum reservoir characterisation – the reservoir property prediction – as an example. We would like to give a typical scenario for the reservoir property prediction to indicate the concrete requirements. In order to identify which components should be contained in a typical petroleum reservoir characterisation system, without loss of generality, consider a geological research institute which

provides the service to predict reservoir property. In such a place, there are an administrator, one personnel officer, some reservoir property prediction experts (decision makers), and several document keepers. The use case of this scenario is presented in Figure 7.1.

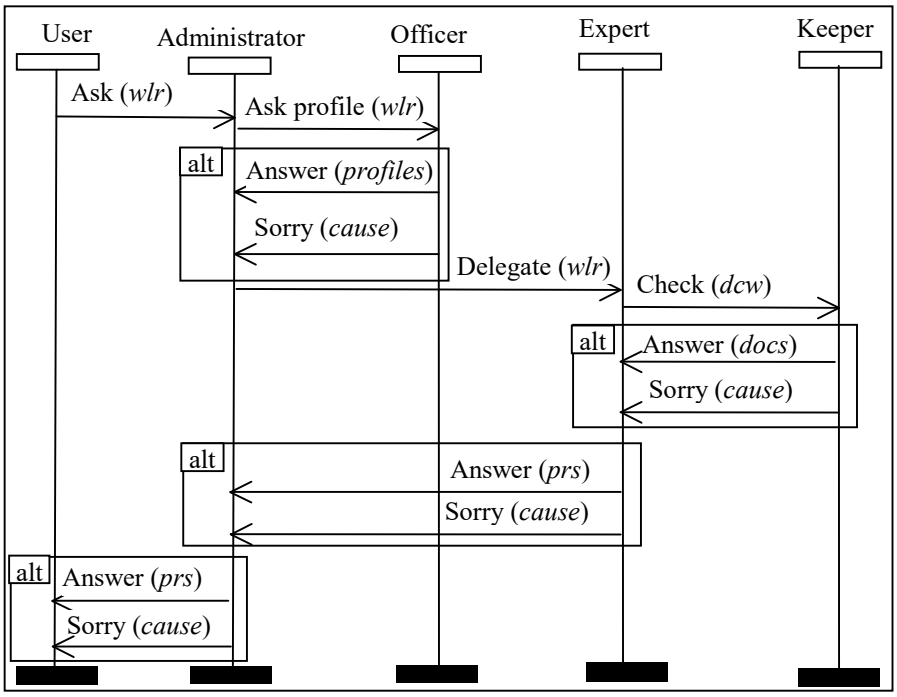

Figure 7.2 MSC process of petroleum reservoir characterisation

The *parameter prediction* process is initiated by a user contacting the administrator with new well logs and a set of requirements (*wlr*). The administrator asks the personnel officer to provide the experts' profiles, and then delegates each reservoir property to one or more experts based on the experts' profiles. The experts then work on the task and try to give their prediction results (*prs*). When the experts make decisions, they may ask document keeper for checking some documents of the cored wells (*dcw*). After the experts finish preparing their prediction results (if one reservoir property was assigned to more than one expert, the prediction results from different experts must be combined to form a final one), they pass it to the administrator. Finally, the administrator sends the prediction results to the user. Such a typical process can be shown in Figure 7.2 with MSC.

7.1.2 Hybrid Integration Strategy Identification

From the petroleum reservoir characterisation requirements, we know that at least four processes need intelligent techniques. They are: well logs curve digitizing (*WLCD*), complicated lithology identification (*LI*), porosity prediction (*PP*), and permeability estimation (*PE*). If one reservoir property can be predicted by more than one intelligent technique, the predicting processes may include more than three in the system. At the same time, a combination process -- ordered weighted averaging (*OWA*) (Zhang and Zhang 2004) -- must be added. The intelligent technique characteristics of those processes are summarised in Table 7.1. The intelligent techniques, namely, neural network, genetic algorithm, combination of fuzzy logic and neural network (FL and NN), and expert system, are selected.

From the requirements and use case of the petroleum reservoir characterisation, two hybrid integration strategies (*MHS* and *AHS*) can be selected. The *MHS* is *loose-coupling* and the *AHS* is *fully-integration*. This is because each intelligent process has loose interactions and some data or information between these intelligent processes may be transmitted by data file.

Table 7.1 The intelligent characteristics of the processes

Process	Characteristics	Intelligent technique
WLCD	if-then rules; inference; fuzzy set	Expert System
LI	Evolutionary computing, random parameter	Genetic Algorithm
PP	Uncertain prediction; matrix data	Neural Network
PE	Fuzzy set; uncertain prediction; similarity degree	FL & NN
OWA	if-then rules; inference; fuzzy set	Expert System

Because the *MHS* is loose-coupling, $\Vdash$ is selected as the hybrid technique relation by following the relation selection mechanism described in Section 4.3.2. So, the loose-coupling hybrid strategy and $\Vdash$ hybrid technique relation meet the requirements of the petroleum reservoir identification system.

7.2 Analysis of the System

Based on the description in Section 7.1, the process organizational relationships, tasks, agents, groups, categories, VOs, dynamics, the intelligent techniques of the system, and the interactions between agents, groups, and categories are discussed in this section.

7.2.1 Organisation Model

The people in the petroleum reservoir characterisation system can be divided into three categories: user, management, and service providers (experts and documents keepers). The member in the management category is the mediator between user and service providers. The organisation of the petroleum reservoir characterisation system is shown in Figure 7.3.

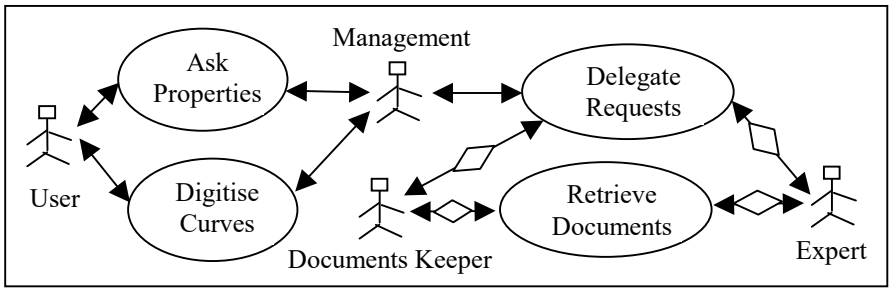

Figure 7.3 Organisation of the petroleum reservoir characterisation

When a user wants to predict the reservoir properties with some well logs, he/she usually goes to the geological research institute for predicting the parameters of reservoir property. The first thing the administrator needs to do is to understand the user's work circumstances (*UWC*). The administrator may ask the user to provide the following information about *UWC*: his/her company, the location of the well, the sondes obtained the well logs (*WL*), etc. Based on the information, the administrator will partition user's requirements as three kinds of tasks: lithology

identification documents (*LID*), porosity prediction documents (*PPD*), and permeability estimation documents (*PED*).

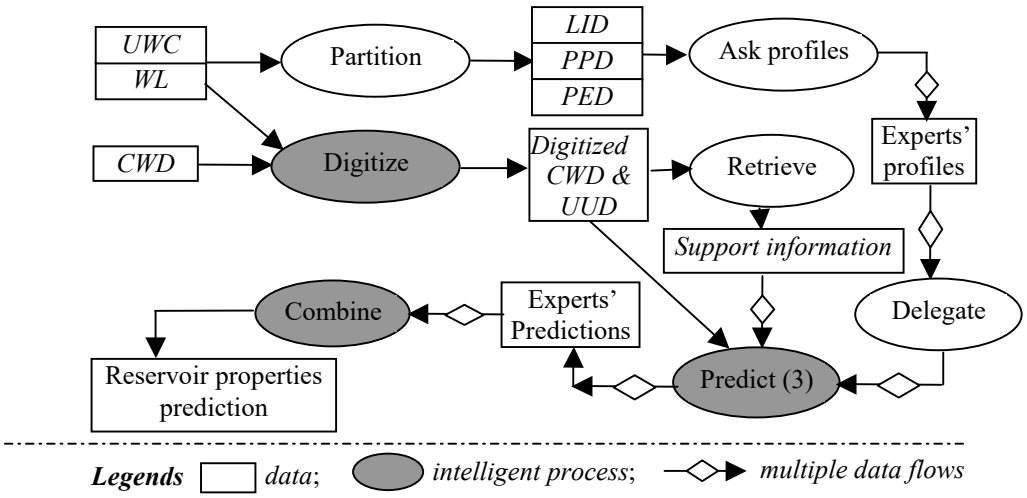

Figure 7.4 DFD of petroleum reservoir characterisation

The administrator asks the personnel officer for available experts with *LID*, *PPD*, and *PED*. When the administrator gets the experts' profiles, he/she will delegate the user's requests to the experts. The experts predict the parameters according to the core well documents (*CWD*), users' uncore well documents (*UUD*), and their knowledge. At last, the administrator presents the results to user, after combining the experts' predictions. According to the descriptions and the MCS process of petroleum reservoir characterisation presented in Figure 7.2, the data flow diagram (DFD) of the system can be obtained as shown in Figure 7.4.

Because loose-coupling is selected as the petroleum reservoir characterisation system hybrid strategy, the process categorising is elective. However, the functionality and interactions must be taken into account.

7.2.2 Task Model

According to the aforementioned descriptions, it is comparatively straightforward to identify the tasks. Figure 7.5 presents the decomposing process. Table 7.2 lists the description of tasks. The process and goal of a task are also presented in this Table.

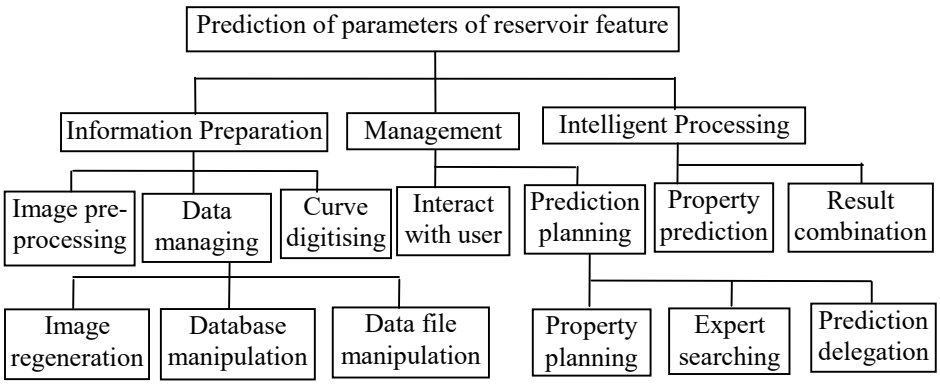

Figure 7.5 The task tree of petroleum reservoir characterisation

Table 7.2 The tasks of petroleum reservoir characterisation

Task Name	Description
Image Pre-processing	Scan well logs as TIFF file; Compress the TIFF file. **P**rocess: Digitising; **G**oal: document digitisation.
Image Regeneration	Redraw well logs based on compressed data or digitised data **P**rocess: Digitising; **G**oal: data transformation.
Curve Digitising	Track and rectify well logs curves based on expertise **P**rocess: Digitize; **G**oal: document digitizing.
Database Manipulation	Control to operate database in agent environment **P**rocess: Retrieving; **G**oal: operate database in agent system
Data File Manipulation	Control to operate data files in agent environment **P**rocess: Retrieving; **G**oal: operate data files in agent system
Interaction With User	Control to input user's requirements and to output results **P**rocess: Partitioning; **G**oal: user interface
Properties Planning	Partition the rock properties and make them can be carried out **P**rocess: Partitioning; **G**oal: each property can be predicted
Expert Searching	Search which expert can do specific prediction task **P**rocess: Asking Profiles; **G**oal: look for suitable expert
Prediction Delegation	Delegate specific prediction task to one or more experts **P**rocess: Delegating; **G**oal: assign a expert to do a prediction
Property Predictions	Complete the predictions according to the experiences **P**rocess: Predicting; **G**oal: predict a reservoir property
Results Combination	Combine the multiple predictions of a property into one **P**rocess: Combining; **G**oal: generate final result

7.2.3 Agent Model

The agent model includes nine kinds of agents as described in Table 7.3.

Table 7.3 The agent model of petroleum reservoir characterisation

Name (number)	Category	Group	Services
Image Pre-processing (1)	Application	Curve Digitising	Carry out Image Pre-processing task
Image Regeneration (1)	Application	Curve Digitising	Carry out Image Regeneration task
Curve Digitising (1)	Service Provider	Curve Digitisation	Carry out Curve Digitising task
Middleware (2)	Service Provider	Middleware Agent	Carry out Database Manipulation and Data File Manipulation tasks
Interface Agent (1)	Application	Property Prediction	Carry out Interaction With User task
Properties Planning (1)	Application	Property Prediction	Carry out Property Planning task
Middle Agent (4)	Middle Agent	Middle Agent	Carry out Expert Searching and Prediction Delegation tasks
Property Predictions (3)	Service Provider	Intelligent Technique	Predict lithology and reservoir properties (porosity and permeability)
Results Combination (1)	Service Provider	Decision Aggregation	Carry out Results Combination task

All the agents in the petroleum reservoir characterisation system are divided into three categories: *application agent*, *middle agent*, and *service provider agent* in correspondence with the user, management, and service provider in the *organisation model*. The *application agent category* includes some user's tasks and part of administrator's tasks (asking profile process in Figure 7.4). The well logs curve digitising and reservoir property prediction in *application agent category* is defined as two agent groups. There are several agents in each agent group. The *middle agent category* includes part of administrator's tasks and all tasks of

personnel officers (delegating process in Figure 7.4). Each group in *middle agent category* includes two middle agents and serves for one responsible group located in *application agent category*.

The *service provider agent category* includes part of administrator's tasks (results combination in Figure 7.4), all tasks of prediction experts, and tasks of document keepers (all intelligent processes in Figure 7.4). The *service provider agent category* includes four agent groups: middleware agent (wrappers of database and data file server, retrieving process in Figure 7.4), reservoir property prediction agent (predicting process in Figure 7.4), curve digitisation agent (digitising process in Figure 7.4), and decision aggregation agent (combining process in Figure 7.4). Each group in *service provider agent category* includes different agents.

Here, we give the middle agent in activity diagram as shown in Figure 7.6. Other agents' graphic representations are omitted for saving space.

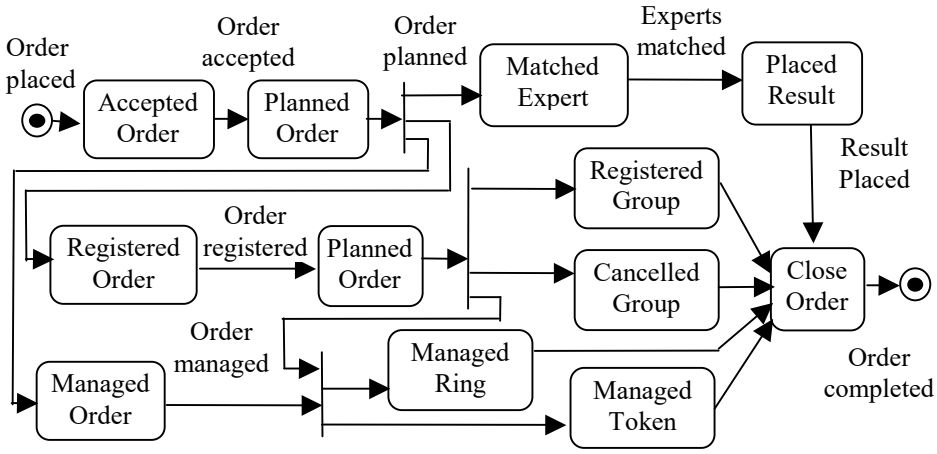

Figure 7.6 Activity diagram for middle agent

7.2.4 Coordination Model

Figure 7.7 shows the interactions of the well logs curve digitising process in MSC. Figure 7.8 shows the interactions of logs regeneration process. Figure 7.9 shows the interactions of reservoir property predicting process. If the number of the prediction

agents for each reservoir property parameter is more than one, the result combination agent should be taken into account in Figure 7.9.

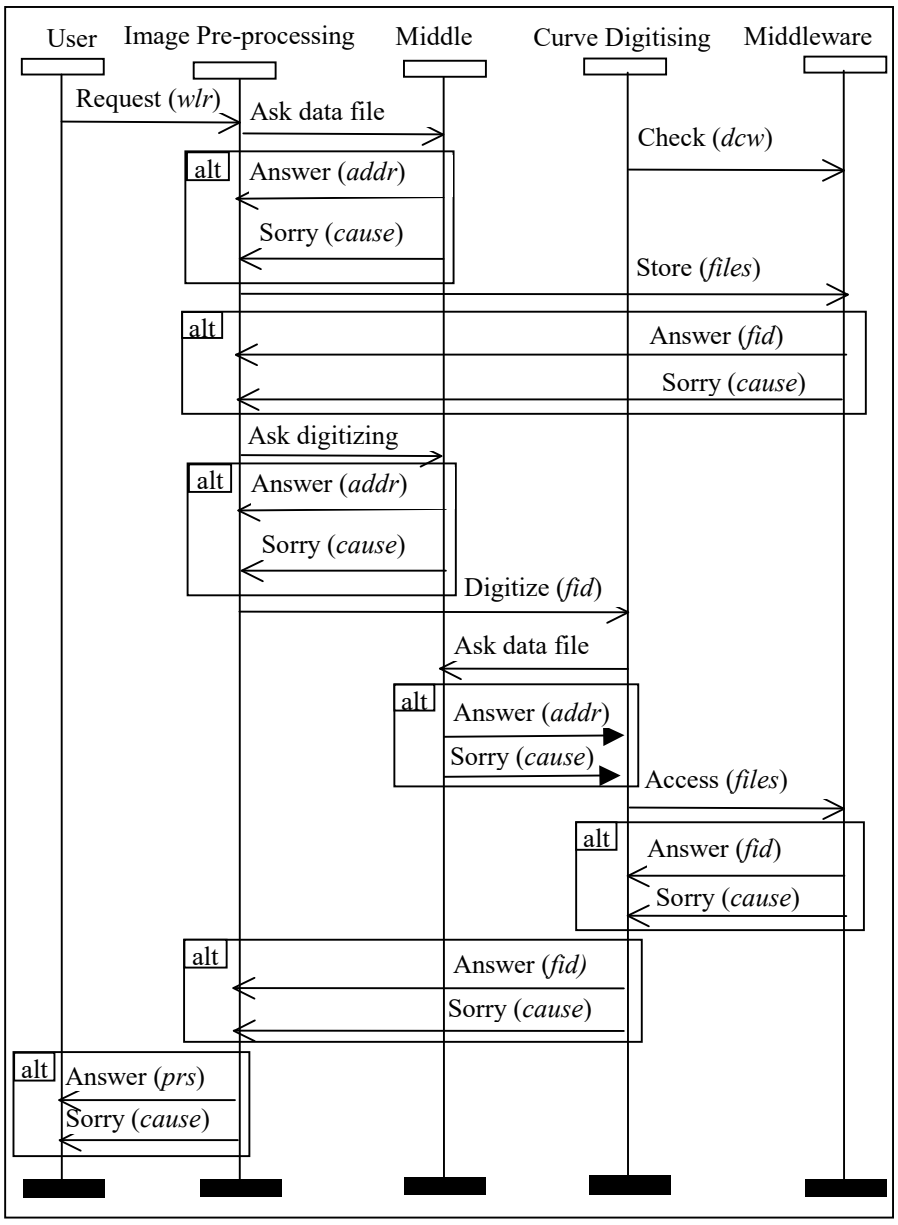

Figure 7.7 MSC process of well logs curve digitising

Figure 7.10 shows the interchanged messages between agents in event flow diagrams. Figure 7.11 shows two typical interactions with the state transition diagrams of SDL.

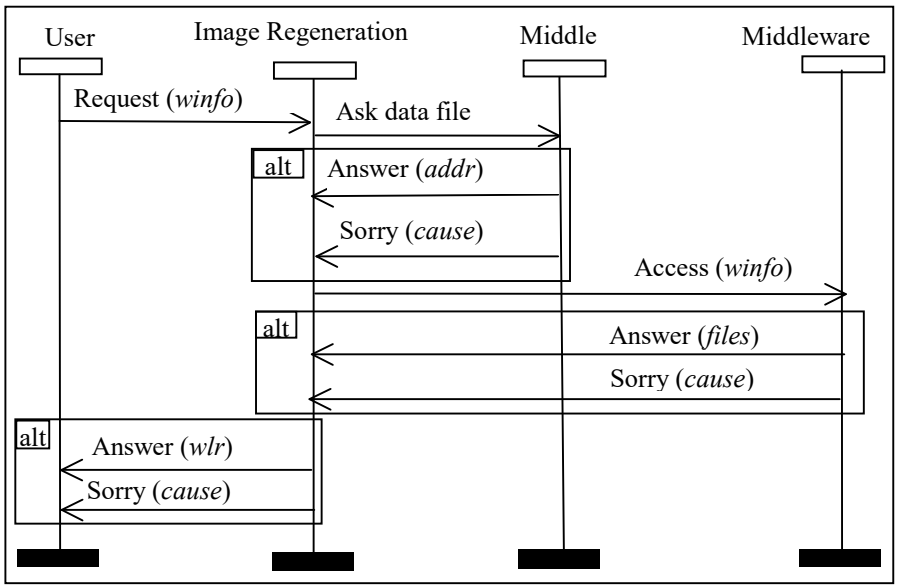

Figure 7.8 MSC process of well logs regeneration

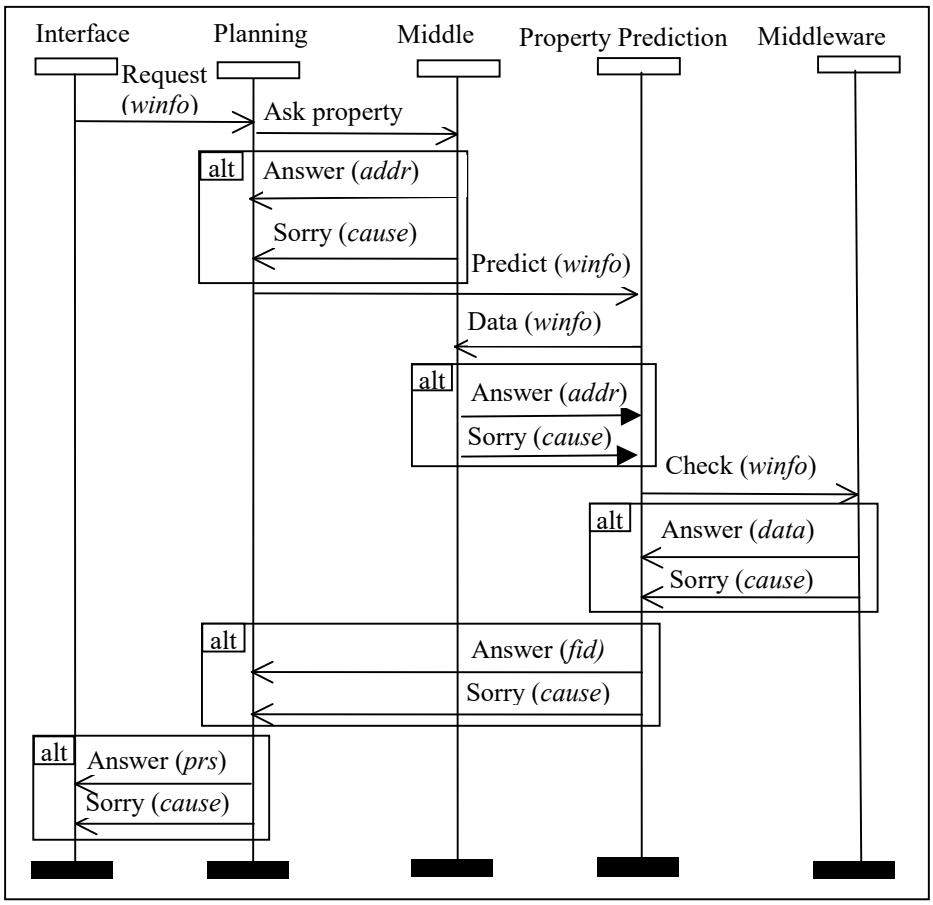

Figure 7.9 MSC process of reservoir property prediction

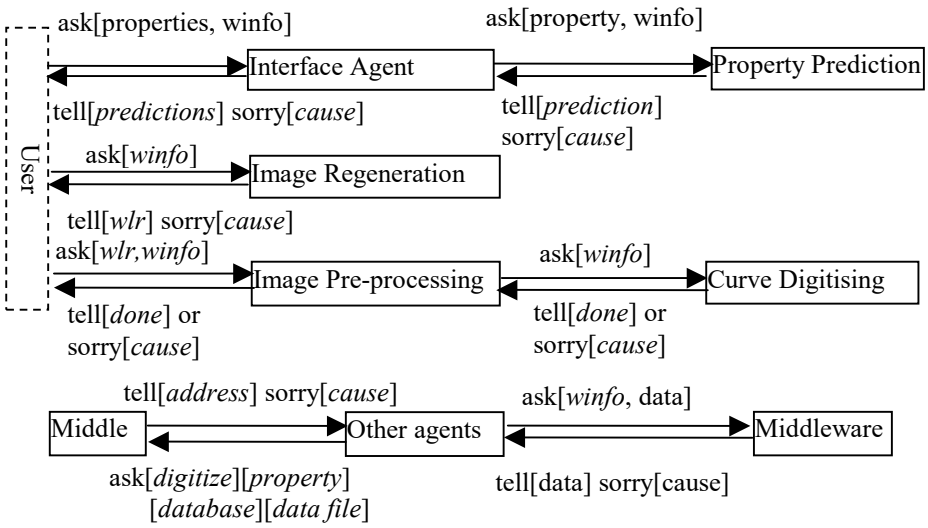

Figure 7.10 Even flow diagram of the system

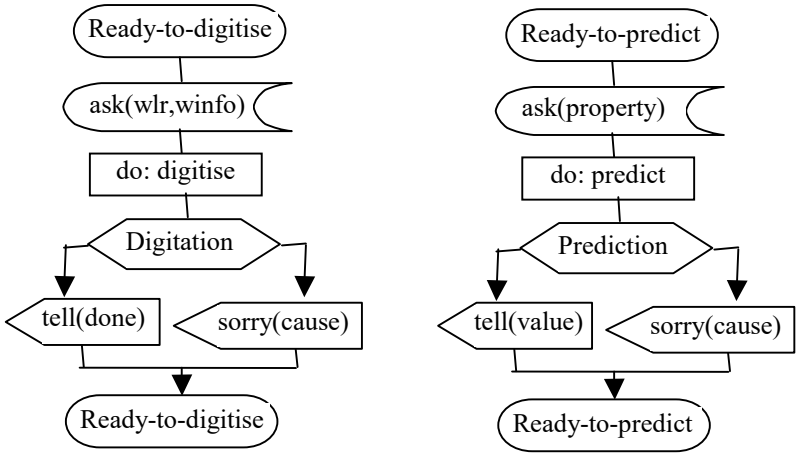

Figure 7.11 Typical interactions with SDL state diagrams

Another interaction, well logs regeneration, is similar with well logs digitising and property prediction. The agents in each interaction must be asynchronous. However, the agents between the three interactions may be synchronous. The flexibility of the system has been considered in PAHIS design. We need not to mention too more here. In petroleum reservoir characterisation system, the agents in a group of the *application agent category* are asynchronous, but agents in the *service provider agent category* are synchronous.

7.2.5 Expertise Model

Four agents (lithology identification, porosity prediction, permeability estimation, and curve digitizing) have employed intelligent techniques. These problem solving methods (PSMs) can be defined by CML.

KNOWLEDGE-MODEL petroleum-reservoir-characterisation;
DOMAIN-KNOWLEDGE reservoir-properties-prediction-domain;
PSM-KNOWLEDGE psm-lithology-identification;
complicated lithology identification with parthenogenetic algorithm;
END PSM-KNOWLEDGE psm-lithology-identification;
PSM-KNOWLEDGE psm-porosity-predication;
porosity prediction with neural network;
END PSM-KNOWLEDGE psm-porosity-predication;
PSM-KNOWLEDGE psm-permeability-estimation;
permeability estimation with fuzzy neural network;
END PSM-KNOWLEDGE psm-permeability-estimation;
PSM-KNOWLEDGE psm-curve-digitising;
well logs curve-digitizing with expert system;
END PSM-KNOWLEDGE psm-curve-digitising;
END DOMAIN-KNOWLEDGE reservoir-properties-prediction-domain;

- ***Complicated lithology identification with parthenogenetic algorithm***

Considering the classification indetermination and diversity of the complicated lithology partition in petroleum reservoir geology, a method based on parthenogenetic algorithm's pattern clustering is proposed. This method realizes evolving operation by genetic operators such as genetic transposition, genetic shift and so on, Using parthenogenetic algorithm to solve pattern clustering problem make clustering result independence of initial clustering.

Step 1: Gene Coding. If there are L pattern samples, number them with 1, 2... L. Allocate each sample number a chromosome randomly and make each chromosome include L genes. That is the chromosome's string length is L.

Step 2: Sample chromosomes clustering. The clustering rule is based on the class distinguishable conditions. If some samples belong to the same class, they have the same class distinguishable conditions. That is, each class has its own

distinguishable conditions. Firstly, if the first i_1 ($i_1 \geq 1$) genes match the first class distinguishable conditions, then put the *i1* genes in the first class. Secondly, if the first i_2 ($i_2 \geq 2$) remaining genes match the second class distinguishable conditions, then put the *i2* genes in the second class. In this way, each gene is put in a suitable class. The definition of class distinguishable conditions depends on a specific problem. We have defined the class distinguishable conditions according to class density in the lithology identification as following:

$$f(i) = \frac{m_i}{d^r}$$

where *f(i)* if the class density of class *i*; m_i is sample number of class *i*; *r* is sample character dimension; *d* is the max distance that sample point x_j of class *i* to sample centre, that is, $d = \max\left\|x_j - x_{0i}\right\|$, *d* is also called class radius.

Class density is called big class density if all *L* samples are in the same class. The big class density is defined as following:

$$T_0 = \frac{L}{d_0}$$

where d_0 is the maximum distance that all *L* sample points to sample centre point x_0, that is, $d_0 = \max\left\|x_k - x_0\right\|$.

The ratio *g* between a class density and the big class density is called class density threshold.

Step 3: Adjudge an individual by adaptive function. Define an adaptive function that can measure an individual whether it is suitable in a specific class. The adaptive function is as following:

$$f(k) = \alpha_1 F_k + \frac{\alpha_2}{s_k}$$

where s_k is the class number of clustered individual *k*; α_1 and α_2 are weights of gene; F_k is the average class density of the individual *k*; that is, $F_k = \frac{\sum_1^{s_k} f_k(i)}{s_k}$; $f_k(\,)$ is the class density function of the individual *k*.

- ***Porosity prediction with neural network***

We adopt Wong's approach (Wong, Gedeon et al. 1995) to predict the porosity of reservoir property. The neural network is composed of three kinds of layers: input, middle and output layers and has the 2-3-3 configuration. The input layer is composed of two nodes, and the middle and output layers contain three nodes. Mathematically speaking, this neural computation involves a transformation of an input vector with two components (X1, X2) into an out vector with three components (Y1, Y2, and Y3). The magnitude of the output vector depends on the weights on all connections. Bias nodes are usually included for faster convergence and better decision boundaries. The weights on these nodes are treated the same as the others, and the input activation values of these connections are always equal to one.

The backpropagation (BP) algorithm is employed. Before beginning training, some small random numbers are used to initialize each weight on each connection. BP requires pre-existing training patterns, and involves a forward-propagation step followed by a backward-propagation step. The forward-propagation step begins by sending the input signals through the nodes of each layer. The sigmoid function is used at each node for the transformation of the incoming signals to an output signal. This process repeats until the signals reach the output layer and an output vector is calculated. The output of node *j* in the middle layer is calculated by:

$$b_j = f(S_j) = \frac{1}{1+e^{-sj}} \quad S_j = \sum_{i=1}^{n} W_{ij} a_i - \theta_j \quad j=1, 2, \ldots, p$$

where *f( )* is the sigmoid function; W_{ij} is the connection weight from node *i* in the input layer to node *j* in the middle layer; θ_j is the threshold of a node; *p* is the node number in middle layer.

The output of node *t* in the output layer is calculated by:

$$C_t = f(L_t) \quad L_t = \sum_{j=1}^{n} V_{jt} \times b_j - \gamma_t \quad t=1, 2, \ldots, q$$

where V_{jt} is connection weight from node j in the middle layer to node t in the output layer; V_t is the threshold of node t in the output layer; q is the node number in the output layer.

The backward-propagation step calculates the error vector by comparing the calculated and target outputs. New set of weights are iteratively calculated, by modifying the existing weights, based on these error values until a minimum overall (global) error is obtained. The error of the output layers is defended as:

$$d_t^k = (y_t^k - C_t^k) f^{'}(L_t) \quad t=1, 2, \ldots, q; k=1, 2, \ldots, m$$

where $(y_t^k - C_t^k)$ is the absolute error.

The error of the middle layers is defended as:

$$e_j^k = [\sum_{i=1}^{z} V_{di}^k] f^{'}(S_j)$$

The weights and thresholds can be adjusted by:

$$\Delta V_{jt} = a \times q d_t^k \times b_j^k \quad j=1, 2, \ldots, p; t=1, 2, \ldots, q$$

$$\Delta V_t = a \rangle d_t \quad k=1, 2, \ldots, m \ (0<\alpha<1)$$

$$\Delta W_{ij} = \beta \times e_j^k \times a_j^k \quad i=1, 2, \ldots, n; j=1, 2, \ldots, p$$

$$\Delta \theta_j = \beta \, e_j^k \quad k=1, 2, \ldots, m \ (0<\beta<1)$$

where α and β are learning rate.

The root-mean-square error (RMSE)is used as a measure of the global error which is defined as

$$RMSE = \sqrt{\frac{\sum_i^{n_p} \sum_j^{n_0} (yij - xij)}{n_p . n_0}}$$

where n_p is the number of input/output pairs making up the training patterns, n0 the number of notes in the output layer, x and y are the output and target signals respectively.

Lithofacies information extracted from core samples is used as the training data set in classification of well log signals because of its high reliability and accuracy compared to well logs. After depth-matching core and log data, the corresponding

well log signals (e.g., gamma ray, sonic travel time and bulk density) can be read for each core sample, and removal of outliers is done. The input data is then normalised in the interval (0, 1). The output data is practically normalised in the interval (0.1, 0.9) for faster convergence. There are no restrictions on how to normalise the data. The whole data set (i.e., the training and validation data) must, however, be normalised in the same manner.

- ***Permeability estimation with fuzzy neural network***

We adopt Huang's approach (Huang, Wong et al. 1996) to estimate the permeability of reservoir property. Normally more than five well logs plus geological facies are available in un-cored wells. Use X to denote well logs and geological facies, and Y to denote permeability. The model is written in the following form:

$$Y = f(\theta(X_0, Y_0), X)$$

where f is a mapping from inputs to outputs, (X_0, Y_0) are the past observation values, and θ is a parameter set depending on (X_0, Y_0).

Four layers (input, fuzzification, hidden, and output) of fuzzy neural network (FNN) has been adopted in this approach. Two inputs, three fuzzy rules, and one output neuron are designed. There are 2x3 neurons in the fuzzification layer. Every neuron in this layer represents a fuzzy membership function for each of the input variables. The activation function used in the fuzzification layer is:

$$A_{ij}(x_j) = \exp(-\left|w_{ij_1} x_j + w_{ij_0}\right|^l)$$

where A_{ij} is the value of radial basis (fuzzy membership) function of the j^{th} input variable corresponding to the i^{th} rule. The variable l is in the range $0.5 \le l \le 5$, and w_{ij} is the connection weight.

The activation function used in the hidden layer is:

$$y_i(x_1, x_2, ..., x_m) = \prod_{j=1}^{m} A_{ij}(x_j)$$

where m is the number of input variables.

The function of the output layer is:

$$Y = \sum_{i=1}^{n} y_i c_i$$

where n is the number of fuzzy rules, c_i is the connection weight. Hence, the rule is of the form:

IF x_1 is A_{i1} and x_2 is A_{i2} and … and x_m is A_{im}

THEN $c_i = b_{i0} + \sum_{j=1}^{m} b_{ij} x_j$ and $Y = \sum_{i=1}^{n} y_i c_i \Big/ \sum_{i=1}^{n} y_i$

where b_{ij} is the connection weight.

- ***Well logs curve-digitizing with expert system***

We have proposed a SCTR (Scanning, Compressing, Tracking, and Rectifying) approach to digitizing well log parameter graph (Li, Song et al. 2003). There are three kinds of rules, namely, image compressing, curve tracking, and data rectifying in well log curves digitizing. The curve tracking and data rectifying is based expert system technique.

✓ **Rules for image compressing**

The purposes of the compressing algorithm include two aspects. Firstly, the redundant message must be filtered out from the image file. Secondly, it is convenient to directly track the curves based on the compressed image files. For solving above problems, we employed LAG data structure to compress the original image file. LAG is a kind of directed graph. Its nodes are all segments of continuous black pixels for all scanning lines of the image; the arcs of a node are the joint relations to other nodes located in its adjacency scanning lines. Assume two nodes A(x_1, x_2) and B(x_3, x_4) which locate in two adjacency lines, respectively. If ($x_2 \geq x_3$) and ($x_1 \leq x_4$) are true, A and B are joint. A is B's parent node and B is A's child node. The structure of a LAG node is shown in Figure 7.12.

X1	*X2*	*Gray*	*Edge1*	*Edge2*	*Offset*

Figure 7.12 Structure of a LAG node

Let *X1* be the start value of the horizontal coordinates of a segment. Let *X2* be the end value of the horizontal coordinates of the same segment. Let *Gray* be the

average gray of the segment. Let *Edge1* be the number of joint nodes located in the previous adjacency line. Let *Edge2* be the number of joint nodes located in the following adjacency line. Let *Offset* be the offset from current node to its first joint node located in the following adjacency line in the LAG link list. The rules for compressing the original TIFF image with LAG are described as the following algorithm.

Input: TIFF dot matrix image data *Im(m,n)*. A line *L(i)* is loaded for each loop of the algorithm, where *L(i)* includes *n* pixels and $0 \leq i \leq m-1$.

Output: *LAG*

Step 1: i=0 and load *L(i)*.

Step 2: Extract the *X1, X2*, and *Gray* of each segment in *L(i)*.

Step 3: Construct all *LAG* nodes in *L(i)*.

Step 4: While ($i < m$-1), do Step 5 to Step 9.

Step 5: i=i+1 and load *L(i)*.

Step 6: Extract the *X1, X2*, and *Gray* of each segment in *L(i)*.

Step 7: Construct all *LAG* nodes in *L(i)*.

Step 8: Fill in *Edge2* and *Offset* (if *Offset* = 0) of related *LAG* nodes in *L(i-1)*.

Step 9: Fill in *Edge1* of *LAG* nodes in *L(i)*.

✓ **Rules for curve tracking**

The goal of curve tracking is to link all nodes in a curve. The rules ae based on the LAG and always attempts to find the successor of current node in the next adjacency line. According to the value of *Edge2* (0, 1, or greater than 1), there are three cases for finding the successor. Figure 7.13 is an example of two cross curves in LAG. The *Edge2* of each segment of part A and D is 1. The *Edge2* of the last segment of part C is 2. The *Edge2* of the last segment of part B and E is 0.

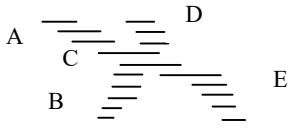

Figure 7.13 Two cross curves in LAG

Curve Segment (CS) data structure is proposed based on LAG for improving the speed and veracity when the curves are traced. A CS is defined as follow: If (*Edge1*≠1) for any LAG node, it is the head of a CS; If (*Edge2*≠1) for that head node or its son LAG node, it is the tail of the CS; All LAG nodes (*Edge1* = 1 and *Edge2* = 1) between the head and the tail (include the head and the tail) are members of the CS. For a specific CS, the number of joint Curve Segments (CSs) located in the previous adjacency line is *Edge1* of its head LAG node; and the number of joint CSs located in the following adjacency line is Edge2 of its tail LAG node. The structure of a CS is shown in Figure 7.14.

<table>
<tr><td>SHN</td><td>STN</td><td>Y</td><td>Depth</td><td rowspan="2">AW</td></tr>
<tr><td>A1</td><td>A2</td><td>SS</td><td>Grid Segment</td></tr>
</table>

Figure 7.14 Structure of a CS

Let SHN be the offset value of the head node of a CS in LAG link list. Let STN be the offset value of the tail node of a CS in LAG link list. Let Y be the value of y coordinates of the head node of a CS. Let Depth be the depth of the CS along y coordinates. Let AW be the average width of the nodes in a CS. Let A1 mean the first attribute of a CS. The value domain of A1 is NOISE, DOT, and LINE. Let A2 mean the second attribute of a CS. The value domain of A2 is *BIG-DOT*, *SHORT-LINE*, and *NEITHER-BIG-SHORT-NOR-SHORT-LINE*. The both above attributes indicate the forms of the curve. Let SS mean segment style of a CS. The value domain of SS is *PUBLIC-SEGMENT* and *PRIVATE-SEGMENT*. Let Grid Segment indicate whether the CS is a centimetre grid.

In order to get the values of A1, A2, SS, and Grid Segment, the following signs are defined:

width: the average width of a CS;

depth: the depth of a CS;

up-edges: the *Edge1* of a CS;

dw-edges: the *Edge2* of a CS.

We propose the following rules for determining the values of A1 (Rule1 to Rule3), A2 (Rule4 to Rule6), SS (Rule7), and Grid Segment (Rule8) by experiment with large numbers of well-logging graphs:

Rule1: If ***width*** ≤ 4 and ***depth*** ≤ 2 and (***up-edges*** =0 or ***dw-edges*** = 0), then *A1 = NOISE*;

Rule2: If ***width*** > 4 and ***width*** ≤ 8 and ***depth*** > 2 and ***depth*** ≤ 8 and (***up-edges*** =0 or ***dw-edges*** = 0), then *A1 = DOT*;

Rule3: Neither Rule 1 nor Rule2, *A1 = LINE*;

Rule4: If *A1=LINE* and ***width*** ≤ 24 and ***depth*** ≤ 24, then *A2 = BIG-DOT*;

Rule5: If *A1=LINE* and ***width*** ≤ 8 and ***depth*** ≤ 40 then *A2 = SHOT-LINE;*

Rule6: If *A1 = LINE* and neither Rule4 nor Rule5, then *A2 = NEITHER-BIG-SHORT-NOR-SHORT-LINE*;

Rule7: If *A1* $\neq$ *NOISE* and *Edge1>1* and *Edge2>1* and ***depth*** >2, then *SS = PUBLIC-SEGMENT*. Otherwise *SS = PRIVATE-SEGMENT*;

Rule8: If the gray of CS satisfies the requirement of grid line, and ***width*** ≤ 2 and ***depth*** ≤ 2, and *SS = PUBLIC-SEGMENT*, then *Grid Segment* = 1. This CS is recognised as grid line. Otherwise, *Grid Segment* = 0.

The tracking algorithm consists of three processes. The first one is to generate all CSs based on LAG, and calculates the attributes of each CS according to Rule1 to Rule8. The second one is to link joint CSs according to their *Edge2* values. The last one is to thin the linked curves. In linking CS process, the following rules are proposed: If *Edge2* equals 1, the successor is the only joint CS located in the following adjacency line. Link the successor to the curve and change the successor as current processing CS until reaching the curve end. If *Edge2* is greater than 1, the successor must be selected according to the characters (e.g. AW, attributes, SS, grey) of the joint CS located in the following adjacency line, or according to the extending trend of the linked curve. If it is still inexplicit, human aid is needed. If *Edge2* is 0 and the depth of well does not reach the desired end, the original curve may be broken. In such case, define a zone that may include the possible successor in the following several lines and find a suitable CS as its successor according to the characters of the CSs and the extending trend of the linked curve (For dot line curve

and broken line curve, the processing methods are similar with this rule). In thinning process, middle axis method is employed. The results are stored in a *CURVEDATA* data structure.

- ✓ **Algorithm for data rectifying**

The digitizing errors of the well log graphs include mismatching two pages (when a graph is generated by connecting two or more pages), graph distortion (because of the environmental factors), and rotating an angle (when the graph is scanned). The centimetre grids on the graph can implicate all above errors, that is, if an error occurs, the related centimetre grids will be correspondingly changed. The distortion rectifying includes two steps: rectify all centimetre grids; and rectify the curves grid by grid in accordance with the rectifying result of each centimetre grid.

7.2.6 Reorganisation Model

The platform of the petroleum reservoir characterisation system can directly employ PAHIS proposed in Chapter 5. However, the categories, groups, and VOs should be defined accordingly. The VO will be automatically defined when an application is run. The member of a VO includes all agents in the application group and relevant intelligent processing agents which are called by the running application.

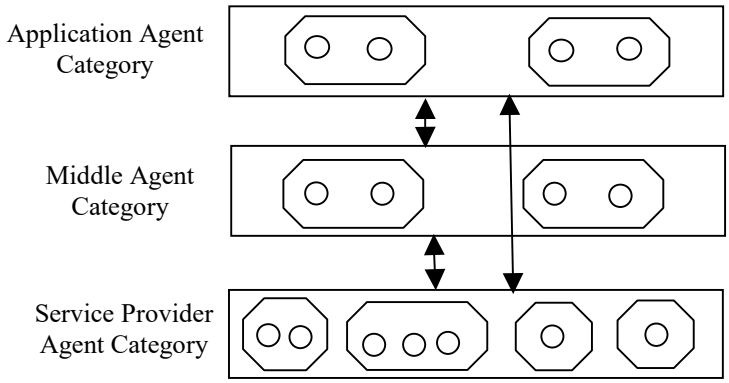

Figure 7.15 The hierarchical structure of the categories

The three categories can be organised with hierarchical structure as shown in Figure 7.15. The well logs curve digitizing and reservoir property prediction in *application agent category* is defined as two agent groups. There are several agents in each agent group. New application based on well logs information and reservoir property prediction intelligent techniques can be added into this category. Each group in *middle agent category* includes two middle agents and serves for one responsible group in *application agent category*. The number of the groups located in the *application agent category* is always same with the one of the groups located in the *middle agent category*. If an application is added into the system, a new group located in the *middle agent category* will be produced automatically. The *service provider agent category* includes four agent groups: middleware agent, reservoir property prediction agent, curve digitisation agent, and decision aggregation agent. Each group located in *service provider agent category* may include different number of agents. New developed service provider agents can be added into the system by means of advertising themselves to PAHIS and indicating which group the agents prefer. That is, the groups in this category are managed by middle agent rather than the *service provider agent category*.

7.3 Design of the System

The design phase consists of four steps: architecture design, agent communication design, platform design, and application design. Because petroleum reservoir characterisation system adopts PAHIS platform, the platform design step is omitted.

The first step to be conducted is the design of the system architecture (Figure 7.16). Because there are two application groups in the petroleum reservoir characterisation system, there are four middle agents (host 1, duplicate 1, host 2, and duplicate 2) in PAHIS in correspondence with the application groups: the *well log curve digitizing* and the *reservoir property prediction*. The middle agents are organised with the ring-based architectural model as discussed in Chapter 5. Certainly, PAHIS supports more than two application groups added into the

platform dynamically. Because middleware agents are used for manipulating database, a database-based agent communication language (DBACL) is proposed by enriching KQML.

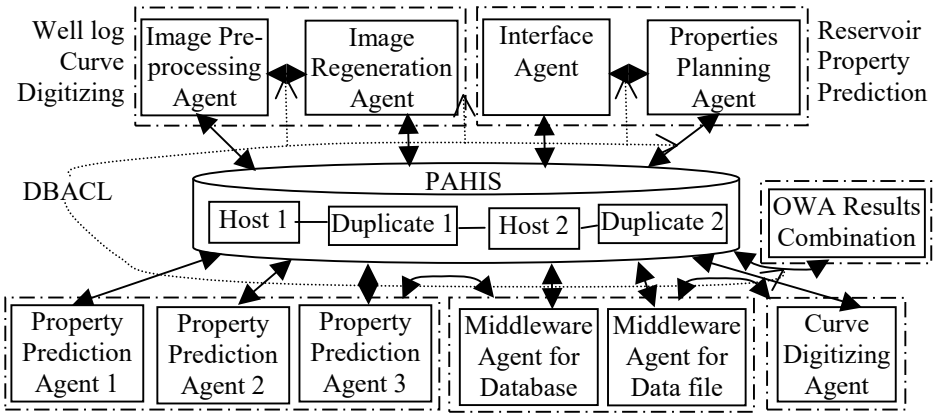

Figure 7.16 Architecture of reservoir characterisation system

The DBACL is proposed by enriching a KQML primitive: *query*. The primitive *query* is defined as:

query

: sender <*DatabaseAccessRequestAgent*>

: receiver <*MiddlewareAgent*>

: content <*DatabaseManipulationExpression*>

: language C

: ontology Agent-database

The format of *content* is as following (Li, Zhang et al. 2002a):

&$ITEM1:PARAMETER1_VALUE$

$ITEM2:PARAMETER2_VALUE$……@

where "&" is the beginning mark of a database operation; "@" is the end mark of the database operation; "$" is the beginning and end marks of an item (statement). Each item includes three parts: parameter, its value and a separator ":". The statements defined in the *content* of query primitive and their meanings and comments are presented in Table 7.4. See (Li, Zhang et al. 2002a) and (Zhang, Li et al. 2002) for details.

Table 7.4 The statements in *content* of query primitive

Items	Meaning	Comments
TP	Request type	1: query data; 2: update data; 3: delete data; 4: input data; 5: create table
TY	Query type	1: query curve data; 2: query others
TNM	Table name	String. If more one, separated by comma
ZNM	Field name	String. If more one, separated by comma
FITP	Data type	N: integer; F: float; C: character; D: date; L: Boolean; M: memo
FWID	Width of field	Point out the width of a field and the location of radix point.
FDOT	Location of radix point	When convert a float number to string, point out the location of radix point
COND	Query condition	If more then one condition, use operator AND, OR, etc.
WNM	Well logs name	When query a well logs, use this name
CNM	Curve name	When query a curve of a well logs, use this name
SDEPTH	Start point	Y-coordinate start value of the curve
EDEPTH	End point	Y-coordinate end value of the curve
PORT	Socket port	Port of machine for communication
ADR	IP address	12 decimal digits
USR	Logon user name	String
PASS	Logon password	String
DUSER	Database logon user name	String
DPASS	Database logon password	String
PATH	User work directory	A string of valid path
FNAME	File name	A string consists of path and file name
RESULT	Success or not	1 Success; 2 Failure

The following are some examples in the *content* of query primitive to operate the database.

Example 1: Query curve data.

&$TP:1$$TY:1$$WNM:*well-logs-name*$$CNM:*curve-name1,curve-name2,…* $$SDEPTH:*start-point*$$EDEPTH:*end-point*$@

Example 2: Query non-curve data.

&$TP:1$$TY:2$$TNM:*table-name*$@

Example 3: Update data.

&$TP:2$$TNM:*table-name*$$ZNM:*fied-name1;filed-name2;…* $$FITP:*field-type1;field-ype2;…*$$FWID:*field-width1,radix-point;field-type2,radix-point;…* $$FNAME:*data-file-name*$$COND:*condition-of-operator*$@

Example 4: Delete data.

&$TP:3$$TNM:*table-name*$$COND:*condition-of-operation*$@

Example 5: Insert data.

&$TP:4$$TNM:*table-name*$$ZNM:*field-name1;field-name2;…* $$FITP:*field-type1;field-type2;…*$$FWID:*field-width1,radix-point; field-type2,radix-point;…* $$FNAME:*data-file-name*$$COND:*confition-of-operation*$@

Example 6: Create new table.

&$TP:5$$DUSER:*logon-user-name-of-database*$$DPASS: *logon-password-of-database*$$TNM:*table-name*$$ZNM:*field-name1;field-name2;…* $$FITP:*fied-type1;field-type2;…*$$FWID:*field-width1,radix-point; field-width2,radix-point;…*$@

There are two applications, well log curve digitizing and reservoir property prediction, in the petroleum reservoir characterisation system. The well log curve digitizing application completes well log curves digitizing and regenerating. The following agents must be employed for completing the two tasks: image pre-processing agent, image regeneration agent, and curve digitizing agent. Figure 7.17 shows the components of the application. Because well log curve digitizing application must interact with users, image pre-processing agent and curve re-generation agent manage the inputting information by keyboard, displaying states of the agent, image scanning, and displaying image on screen. The image pre-processing agent controls to scan well logs parameter graph into computer as image files (TIFF format) and save the image files as temporary image files to temporary file hard disk (TF). Some images may need to be pre-processed for forming a standard image pattern before compressing and digitizing. And then the agent controls to compress standard image pattern files to data files, which will be stored forever in hard disk. At the same time, a large amount of hard disk space is saved. The curve digitizing agent digitizes the compressed image files (by tracking and rectifying). The image re-generation agent manages redrawing the well logs based on the compressed data files or digitized well log data.

The reservoir property prediction application completes lithology identification, porosity prediction, and permeability estimation. The following agents must be employed for completing the three tasks: lithology identification agent, porosity prediction agent, permeability estimation agent. Figure 7.18 shows the components

of the petroleum reservoir prediction application. The intelligent technique agents employ the intelligent techniques described in Subsection 7.2.5.

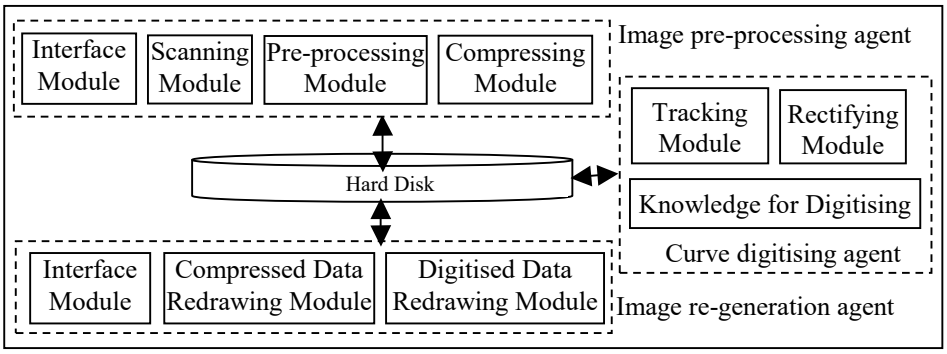

Figure 7.17 Components of well log digitising application

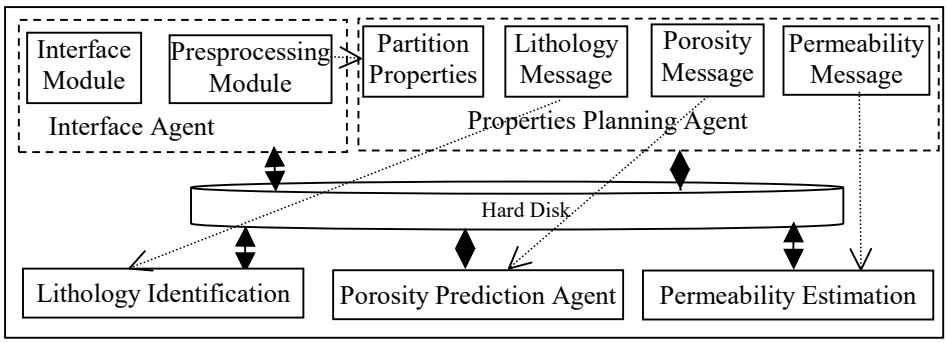

Figure 7.18 Components of reservoir property prediction application

7.4 Implementation of the System

Under the support of PAHIS, the prototype of the petroleum reservoir characterisation system has been implemented by using C language and Socket technique. Some ideas from "the agent-based framework for petroleum information services from distributed heterogeneous data resources" described in (Li, Wang et al. 2002, Zhang, Li et al. 2002, Li, Zhang et al. 2002a, Li, Zhang et al. 2003d, Li,

Cheng et al. 2004, Li, Liu et al. 2004) have been borrowed to propose and develop all the agents.

When the platform of the system is initiated, all service provider agents and the application groups must register themselves to the platform first. The planning agent in the reservoir property prediction group has its own domain-specific knowledge base as well as meta-knowledge about when to use intelligent technique agents. The middle agent records the capabilities, ontology, and names, group, etc. of all the service provider agents in a multi-agent system. The scenario goes as follows.

After the graph with well log curves is scanned as an image file by scanner, the image file is pre-processed as the standard pattern file and then compressed to characteristic data files, which are saved to data file servers by data file middleware agent. The curves implied in those data files can be digitized to curve data. Transmit curve data to middleware agent by means of DBACL to store the data to Oracle database. The re-generation agent can access curve data or other information in Oracle database or data files by means of middleware agents and redraw the well logs graph again. The agents worked well by processing test with thousands of well logs graphs.

About the reservoir property prediction, we processed 118 stratum samples located in the complicated block (Daqing oilfield, China). Table 7.5 is part of the results related to lithology classification.

Table 7.5 Complicated lithology stratum classification result

Class	Practical lithology	Sampling stratums	Matched stratums	Precision
1	mudstone	24	22	0.928
2	siltstone	18	15	0.833
3	Argillaceous sandstone	21	15	0.714
4	shale	18	15	0.833
5	sandstone	27	25	0.936

7.5 Summary

The petroleum reservoir characterisation system has developed by following MAHIS process stages. In conceptualisation stage, the requirements of the identification of lithology and the prediction of permeability and porosity are presented. At the same time, the hybrid strategy and intelligent techniques have been selected. In the analysis stage, the process organizational relationships, tasks, agents, groups, categories, VOs, interactions, and intelligent techniques have been clarified. The system's architecture, DBACL, and the components of applications have discussed in design stage. The petroleum reservoir characterization system has implemented based on PAHIS.

The Petroleum reservoir characterisation system has the following features: (1) the agents in application groups can easily access all parameter prediction agents, curve digitising agent, and middleware agents; (2) the agents in the service provider category can be added to or removed from the system dynamically; (3) the presence of PAHIS in this system supports the VO concept and makes the agents in the application category and service provider category reusable; (4) the legacy systems or resources can be used in agent environment by means of middleware agents; (5) Overall system robustness is facilitated through the use of the redundant middle agents with ring-based architectural model; (6) a database-oriented agent communication language is adopted by defining the *content* parameter of KQML.

The successful case studies have shown that MAHIS can function properly in the construction of agent-based HIS.

Chapter 8

8 Evaluation of MAHIS

Although the competence of MAHIS can be verified by numerous case studies, it cannot imagine enumerating all the capabilities of MAHIS. On the other hand, it should be evaluated what are the merits and limitations of MAHIS. For clarifying the competence, merits, and limitations of MAHIS, we apply the enriched *attributes tree* to evaluate MAHIS in this chapter. The a*ttributes tree* organises the found criteria in weighted branches. The framework is for qualitative analysis followed by a quantitative rating. After rating the leaves, the value of the root can be calculated. The competence, merits and limitations of MAHIS can be clarified by analysis of these evaluation values. At the same time, MAHIS can be compared to other methodologies. Section 8.1 introduces the framework for agent-oriented methodology (AOM) evaluation. The framework which is proposed by Ceruzzi and Rossi (Cernuzzi and Rossi 2002) is improved in the evaluation of qualitative and quantitative attributes. We present it here for easy understanding of the *attributes tree* and rating methods. Section 8.2 discusses the enriched *attributes tree*. Section 8.3 compares MAHIS with Gaia and MAS-CommonKADS. Section 8.4 analyses the usability of MAHIS for constructing agent-based HIS based on the evaluation results.

8.1 Framework for AOM Evaluation

The framework proposed by Ceruzzi and Rossi is selected to evaluate MAHIS based on the following reasons. Firstly, methodologies can be evaluated qualitatively based on the quantitative criteria because this framework is known as a qualitative analysis followed by a quantitative rating. Secondly, the extension of the evaluation criteria is convenient. The *attributes tree* is constructed in the framework. The attributes tree organises the found criteria in weight branches. To extend the *attributes tree* just adds some defined roots or braches. Finally, this framework is cited by some valuable evaluation frameworks (Dam and Winikoff 2003, Sturm and Shehory 2003, Sudeikat, Braubach et al. 2004).

The Framework consists of six steps (Cernuzzi and Rossi 2002).

Step 1. Application of the paradigm Goal-Question-Metric (GQM). The main objective of this stage is to determine exactly what is needed to be measured and which criteria to take into account to reach the prospective objectives.

Step 2. Specification of an attributes tree. Beginning with the results of the *GQM*, an *attributes tree* is created (see section 8.2). The objective of creating an *attributes tree* is to identify the more general criteria and then to specialise them into finer criteria to obtain a set of quantifiable ones. So, it is possible to apply numeric measures (measurements) of all criteria to reach the tree's root. This model is the base for measurement in later phases.

Step 3. Definition of the empiric relationships and evaluation of qualitative and quantitative attributes. An *attribute tree* is defined and direct or indirect evaluations measurements are carried out. The observations may correspond to empiric relationships among attributes, qualitative evaluations or quantitative evaluations, depending on the criterion and the type of measurement needed. It may be useful to remember that only leaves of the *attributes tree* are evaluated directly. The other values are obtained by indirect observations. Hereinafter a guideline (set of rules) for assigning numeric values of each directly valuable measurable attribute is presented.

For each attribute A_i, a variable X_i is associated taking a real value, i.e., the measured value. Normally, the possible result of the evaluation may be continuous (ranging from 0 to 1), discrete, absolute, or average according to the attribute. In the discrete case the rule assigns to the method value 1 if it meets with the attribute; value 0.8 if it more partially meets with the attribute; value 0.5 if it partially meets with the attribute; value 0.3 if it less partially meets with the attribute; and value 0 (zero) if it does not meet with the attribute. In the absolute case, the amount of items of the attribute is observed. In the average case a formula like the one below is used.

$$F = \frac{\sum_{i=1}^{N} f(i)}{N}$$

where N corresponds to the total amount of items of the observable attribute; $f(i) = 0$ if the method does not meet with attribute i; $f(i)=0.3$ if the method less partially meets with the attribute i; $f(i)=0.5$ if the method partially meets with the attribute i; $f(i)=0.8$ if the method more partially meets with the attribute i; $f(i)=1$ if the method meets with attribute i; F is the average of attributes presented by the method.

Step 4. Definition of a normalised scale type and rules to carry out the mapping: In this step, the objective is to define a normalised scale type and a group of rules for mapping the results obtained in Step 3 as they relate to the normalised numeric scale.

To each attribute numeric values may be assigned, in the range of 0 to 10 that may be mapped according to the following rules (the mapping rules should preserve the representation condition):

- If the possible values range from 0 to 1 then they are multiplied by 10.
- If the result is discrete, the value 0 is assigned to that which does not meet the attribute and 10 to that which does.
- For those measurements that are carried out starting from absolute values (count) and mathematical formulas, the simple rule of the following formula is applied:

$$M = \frac{V_n \times 10}{May}$$

where M represents the mapping result; V_n represents the previously evaluated value; *May* represents the maximum evaluated value between methods.

- For those measurement that present inverse results (greater value implies worst behaviour), the inverse simple rule of the following formula is applied:

$$A = \frac{Men \times 10}{V_n}$$

where A represents the mapping result; V_n represents the previously evaluated value; *Men* represents the minimum evaluated value between methods.

Step 5. Mapping of the relationships to the normalised numeric scale. This allows designers and evaluators to apply stepwise aggregation mechanism in order to obtain an indicator of more general (indirect) attributes for each competitive modelling method or for a single method.

Step 6. Indirect measurement of other attributes and analysis of the results. Starting from the indirect measurement it is possible to infer a latent variable attribute or a more general variable attribute, not directly observable, obtained from the measurement of defined attributes. In the attributes tree each attribute that is not a leaf of the tree may be measured by means of indirect measurement applying the average of immediately direct sub-attributes. Other aggregation mechanisms, like the Logic Scoring of Preference, may be used.

8.2 Attributes Tree

The evaluation framework proposed by Cernuzzi and Rossi is enriched with the *reorganisation attributes* according to the characteristics of HIS. In the extended evaluation framework, the attributes are grouped into four different categories: those concerning the own characteristics of agents (*internal attributes*), those referred to the interaction process (*interaction attributes*), those standing for the reorganisation of agents (*reorganisation attributes*), and those more directly inherent to the design and development process (*other process requirements*). Figure 8.1 shows the hierarchical structure of the enriched *attributes tree*.

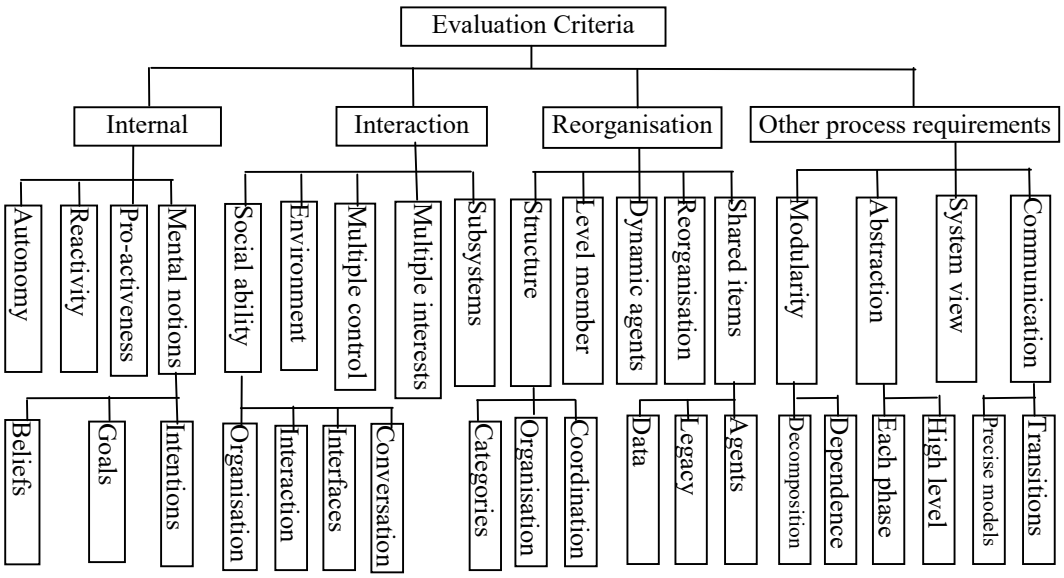

Figure 8.1 Hierarchical structure of the enriched attributes tree

The *reorganisation attributes* include: *hierarchical structure*, *agent as a level member*, *dynamic agents in each level*, *application-based reorganisation*, and *shared items management* which evaluate the capability of methodologies in constructing agent-based HIS.

A. Internal attributes (***average;*** means that the evaluation type is average.)

- *Autonomy (discrete)*: agents encapsulate some states, and make decisions based on this state and its own objectives. So, they have control both over their internal state and over their own behaviour.
- *Reactivity (discrete)*: agents are able to respond in a timely fashion to changes that occur in their environment.
- *Pro-activeness (discrete)*: agents are able to act in anticipation of future goals by taking the initiative.
- *Mental notions (average)*
  - ✓ *Beliefs (discrete)*: agents have to keep information about the environment, the internal state that may hold and the actions it may perform.
  - ✓ *Goals (discrete)*: agents may adopt a set of goals (or desires) that may depend on the actual internal state.

- ✓ *Intentions (discrete)*: agents may have plans they may possibly employ to achieve their goals or respond to events they perceive.

B. Interaction attributes *(average)*

- *Social ability (average)*
  - ✓ *Organisational relationships among agents (discrete)*: when agents interact there is typically some underpinning organisational context that defines the nature of relationships between agents and influences their behaviour. This context may change during the agent's life thus it is important to support simple modifiability to the model.
  - ✓ *Interaction with others agents (average)*: may be necessary either to achieve their individual goals or to manage the organisational dependencies.
    - ➢ *Types of agent's interaction (discrete)*: may vary from information interchanges, to perform a particular action, to co-operation and negotiation or competition, etc.
    - ➢ *Commitments (discrete)*: agents have obligations and authorisations about their relationships with others agents.
  - ✓ *Interfaces with other entities (discrete)*: agents may operate in a more general system composed by other types of entities so it is a need to specify well-defined interfaces.
  - ✓ *Conversations with other agents (discrete)*: different type of agents' interaction (e.g. negotiation, co-operation, etc.) implies a conversation process and therefore requires some kind of agent-communication language. It is important to capture the conversational messages and to facilitate the identification of conversational protocols used in communication.
- *Interaction with the environment (discrete)*: agents are situated in a particular dynamic environment; they receive inputs related to the state of their environment and they may modify the environment through effectors.
- *Multiple Control (discrete)*: interaction between multiple agents implies the administration of multiple loci of control.
- *Multiple Interests (discrete)*: since agents make decisions at run-time, the goal that a specific agent wants to achieve may co-operate, be independent, or enter

in conflict with the goals of other agents in the environment. The administration of multiple interests is imperative.

- *Subsystems interaction (discrete)*: agents may be grouped together into subsystems that may interact between themselves. The interactions within subsystems may be covered by the *Social ability* attributes.

C. Reorganisation attributes *(average)*

- *Hierarchical structure (average)*
  - ✓ *Agent categories (discrete)*: consists of a set of isomorphic agents. Agents in a developed system can be grouped into several categories. A category is different from an organisation. The goal of a category is for easy managing agents rather than for cooperating to complete a task.
  - ✓ *Organisational structure (discrete)*: organises all agents together. There are different performances for different organisational structures. The primitive member in an organisational structure may be an agent or a group (a category of agents).
  - ✓ *Coordination mechanism (discrete)*: decides how agents or agent groups interact under the organisational structure. The interaction protocols of a specific architectural model can be developed. The architectural model in an agent-based system consists of organisational structure and coordination mechanism.
- *Agent as a level member (discrete)*: indicates the type of the primitive member of a level in the hierarchical structure. The possible type may be a single agent or an agent group.
- *Dynamic agents in each level (discrete)*: indicates the mechanism that the primitive member of a level in the hierarchical structure can be dynamically added or removed in running time.
- *Application-based reorganisation (discrete)*: indicates some agents in different categories can be dynamically organised as an organisation. The agents in an organisation can cooperate to complete a specific task. An agent can belong to different organisations at a time. If necessary, an organisation can be dynamically removed.

- S*hared items management (average)*: indicates the mechanism to share the resources in an agent-based system. The resources may include:
  - ✓ *Data (discrete)*. The data source may include databases, data files, websites, etc.
  - ✓ *Legacy systems (discrete)*. Some application systems are not based on agent techniques.
  - ✓ *Agents (discrete)*. Some resources may be organised by agent techniques. To share those resources means to share those agents. These agents are called middleware agents.

D. Other process requirements *(average)*

- *Modularity (average)*: increases efficiency of task execution, reduces communication overhead and usually enables high flexibility. It implies constraints on inter-module communication.
  - ✓ *Decomposition (discrete):* the most basic technique for tackling complexity is to divide the large problem into smaller and more manageable parts each of which can then be dealt with in relative isolation.
  - ✓ *Models' dependence (absolute)*: it is the average of all the relationships between the different models of the modelling method. A high dependence on some specific models of a modelling method may imply that if they are not well designed it may affect all the design; hence, lower dependence is better.
- *Abstraction (average)*
  - ✓ *Abstraction inside each phase (discrete)*: the methodologies present different stages; each stage uses defined models that take into consideration aspects that affect exclusively this stage.
  - ✓ *Existence of design primitives and high level abstraction mechanisms (discrete)*
- *System view (discrete)*: in order to understand the whole system, a macroscopic system-oriented model is required.
- *Communication support (average)*
  - ✓ *Clear and precise models (discrete)*

- ✓ *Systematic transitions (discrete)*: a good modelling method should provide guidelines for simple and elegant transitions between the models.

8.3 Comparison of Methodologies

In this section, MAHIS is compared to MAS-CommonKADS (Mc for short) and Gaia methodologies following the framework described in Section 8.1 and 8.2. There are two reasons to select MAS-CommonKADS and Gaia. Firstly, these two methodologies took the first and second places, respectively, for suitably constructing agent-based HIS according to the ranking results in Section 3.4. Secondly, these two methodologies are based on different technologies: knowledge engineering and object oriented technology. At the same time, these two methodologies are complete and well-grounded.

A. Internal attributes *Gaia: 7.83; Mc: 8.33; MAHIS; 9.58*

A.1. Autonomy

Gaia: the role model, agent model, and service model cover this aspect. *Evaluation 1; Mapping 10* (*Mapping 10* means that the mapping of the result to the normalised ratio scale is 10)

Mc: through the task and agent models the methodology coves this aspect. *Evaluation 1; Mapping 10*

MAHIS: all above models of the Mc have been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

A.2. Reactivity

Gaia: the role model with the interaction model covers this aspect. *Evaluation 1; Mapping 10*

Mc: is specified in the task model. *Evaluation 1; Mapping 10*

MAHIS: the task model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

A.3. Pro-activeness

Gaia: the knowledge model less partially covers this aspect. In effect, it is not possible to specify how to dynamically assume different objectives. *Evaluation 0.3; Mapping 3*

Mc: it is possible to model the goals but not the fuzzy and subjective ones, as well as the evolutionary behaviour. *Evaluation 0.5; Mapping 5*

MAHIS: the fuzzy and subjective goals can be modelled after the system is running. The reorganisation ability can cover this aspect. *Evaluation 1; Mapping 10*

A.4. Mental notions Gaia: 8.33; Mc: 8.33; MAHIS; 8.33

A.4.1. Beliefs

Gaia: the knowledge model partially covers this aspect. It is not possible to model the fuzzy and subjective beliefs. *Evaluation 0.5; Mapping 5*

Mc: it is covered by the expertise model; however it is impossible to model the fuzzy and subjective beliefs. *Evaluation 0.5; Mapping 5*

MAHIS: no improvement comparing with Mc in this aspect. *Evaluation 0.5; Mapping 5*

A.4.2. Goals (Desires)

Gaia: role model covers this aspect. *Evaluation 1; Mapping 10*

Mc: through the task model this aspect is covered. *Evaluation 1; Mapping 10*

MAHIS: the task model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

A.4.3. Actions (Intentions)

Gaia: role model describes the actions and conditions. *Evaluation 1; Mapping 10*

Mc: the task model describes the actions and the methods of problem solving that the agent may adopt for each goal. *Evaluation 1; Mapping 10*

MAHIS: the task model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

B. Interaction attributes *Gaia: 4.7; Mc: 7; MAHIS; 8*

B.1. Social ability

B.1.1. Organisational relationships among agents

Gaia: the agent type concept in agent model more partly covers this aspect. *Evaluation 0.8; Mapping 8*

Mc: the organisation model covers this aspect. *Evaluation 1; Mapping 10*

MAHIS: the organisation model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

B.1.2. Interaction with other agents

B.1.2.1. Types of agents interaction *Gaia: 10; Mc: 10; MAHIS: 10*

Gaia: the interaction model specifies the messages and their order. *Evaluation 1; Mapping 10*

Mc: the coordination model covers satisfactorily the agent interaction. *Evaluation 1; Mapping 10*

MAHIS: the coordination model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

B.1.2.2. Commitments

Gaia: it identifies responsibilities and services of each agent type. *Evaluation 1; Mapping 10*

Mc: the coordination model specifies the conversations among agents, and starting from there, the operations and services. *Evaluation 1; Mapping 10*

MAHIS: the coordination model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

B.1.3. Conversations with other agents

Gaia: the interaction model specifies the messages and their order. *Evaluation 1; Mapping 10*

Mc: the coordination model specifies the conversations among agents. *Evaluation 1; Mapping 10*

MAHIS: the coordination model of the Mc has been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

B.1.4. Interfaces with other entities

Gaia: the service and role models partially cover this aspect. *Evaluation 0.5; Mapping 5*

Mc: the organisation model presents the agents relationships with other objects of the system. *Evaluation 1; Mapping 10*

MAHIS: the organisation model covers this aspect. *Evaluation 1; Mapping 10*

B.2. Interaction with the environment

Gaia: agents may know the environment through their sensors and may react according to the stimuli they receive. However, it is not possible to model changes in the beliefs. *Evaluation 0.5; Mapping 5*

Mc: the expertise model partially covers this aspect. *Evaluation 0.5; Mapping 5*

MAHIS: no improvements comparing with the Mc in this aspect. *Evaluation 0.5; Mapping 5*

B.3. Multiple control

Gaia: the interaction model covers static aspects, not so the dynamic ones. *Evaluation 0.5; Mapping 5*

Mc: the coordination model covers static aspects, not so the dynamic ones. *Evaluation 0.5; Mapping 5*

MAHIS: based on the coordination model of the Mc, reorganisation model covers the dynamic aspect. *Evaluation 1; Mapping 10*

B.4. Multiple interests

Gaia: the acquaintance model partially meets the attributes. *Evaluation 0.5; Mapping 5*

Mc: in the expertise model autonomous and co-operative problem solving methods may be distinguished. The latter partially meets the attributes. *Evaluation 0.5; Mapping 5*

MAHIS: no improvements comparing with the Mc in this aspect. *Evaluation 0.5; Mapping 5*

B.5. Subsystem interaction

Gaia: agent type hierarchy relationships are modelled. However, it does not cover interaction with non-agent systems. *Evaluation 0; Mapping 0*

Mc: the design and organisation models satisfactorily cover this aspect. *Evaluation 1*

MAHIS: the design and organisation models of the Mc have been inherited. MAHIS covers this aspect. *Evaluation 1; Mapping 10*

C. Reorganisation attributes *Gaia: 0; Mc: 1.33; MAHIS; 9.53*

C.1. Hierarchical structure Gaia: 0; Mc: 6.67; MAHIS; 10

C.1.1. Agent categories

Gaia: although the agent type concept supports this aspect, no further description covers this. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model covers this aspect. *Evaluation 1; Mapping 10*

C.1.2. Organisational structure

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: design model covers this aspect. *Evaluation 1; Mapping 10*

MAHIS: design model covers this aspect. *Evaluation 1; Mapping 10*

C.1.3. Coordination mechanism

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: coordination and design models cover this aspect. *Evaluation 1; Mapping 10*

MAHIS: coordination and design models cover this aspect. *Evaluation 1; Mapping 10*

C.2. Agent as a level member

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model covers this aspect. *Evaluation 1; Mapping 10*

C.3. Dynamic agents in each level

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model covers this aspect. *Evaluation 1; Mapping 10*

C.4. Application-based reorganisation

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model covers this aspect. *Evaluation 1; Mapping 10*

C.5. Shared items management Gaia: 0; Mc: 0; MAHIS; 7.67

C.5.1. Data

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model covers this aspect. *Evaluation 1; Mapping 10*

C.5.2. Legacy systems

Gaia: no model covers this aspect. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model can describe this aspect; however, the design is not mentioned. *Evaluation 0.3; Mapping 3*

C.5.3. Agents

Gaia: no model covers his aspect. *Evaluation 0; Mapping 0*

Mc: no model covers this aspect. *Evaluation 0; Mapping 0*

MAHIS: reorganisation model covers this aspect. *Evaluation 1; Mapping 10*

D. Other process requirements *Gaia: 7.5; Mc: 9.5; MAHIS; 9.5*

D.1. Modularity Gaia: 10; Mc: 8.2; MAHIS; 7.85

D.1.1. Decomposition

Gaia: different models facilitate an intuitive use of this aspect. *Evaluation 1; Mapping 10*

Mc: different models cover this aspect. *Evaluation 1; Mapping 10*

MAHIS: all models cover this aspect. *Evaluation 1; Mapping 10*

D.1.2. Model's dependence

Gaia: considering 5 models proposed and the 5 relationships, the average dependence (corresponding to the evaluation results) is *1; Mapping 10*

Mc: considering the 7 models proposed and the 11 dependence relationships, the average dependence is *1.57; Mapping 6.4*

MAHIS: considering the 8 models proposed and the 14 dependence relationships, the average dependence is *1.75; Mapping 5.7*

D.2. Abstract Gaia: 10; Mc: 10; MAHIS; 10

D.2.1. Abstraction inside each phase

Gaia: it contemplates abstraction levels in different phases. *Evaluation 1; Mapping 10*

Mc: the first phases contemplate a higher abstraction level which is easily refined in design phase. *Evaluation 1; Mapping 10*

MAHIS: it contemplates abstraction levels in different phases. *Evaluation 1; Mapping 10*

D.2.2. Existence of design primitives and high level abstraction mechanisms

Gaia: it covers these aspects. *Evaluation 1; Mapping 10*

Mc: it covers these aspects. *Evaluation 1; Mapping 10*

MAHIS: it covers these aspects. *Evaluation 1; Mapping 10*

D.3. System view: macroscopic system-oriented model

Gaia: it does not cover this aspect. *Evaluation 0; Mapping 0*

Mc: the organisation model offers a global view of the system through the application design. *Evaluation 1; Mapping 10*

MAHIS: the organisation model includes this aspect. *Evaluation 1; Mapping 10*

D.4. Communication support Gaia: 10; Mc: 10; MAHIS; 10

D.4.1 Clear and precise models

Gaia: it satisfactorily covers this aspect. *Evaluation 1; Mapping 10*

Mc: it satisfactorily covers this aspect. *Evaluation 1; Mapping 10*

MAHIS: it satisfactorily covers this aspect. *Evaluation 1; Mapping 10*

D.4.2. Systematic transitions

Gaia: the methodology offers simply transition mechanisms from higher abstraction levels up to the design and the implementation. *Evaluation 1; Mapping 10*

Mc: the methodology offers simply transition mechanisms from higher abstraction levels up to the design and the implementation. *Evaluation 1; Mapping 10*

MAHIS: inherit the characteristics of the Mc. *Evaluation 1; Mapping 10*

Table 8.1 shows the mapping of the results to the normalised ratio scale as defined in step 4. It is possible to observe that the attributes obtained by indirect measurement were obtained by averaging the attributes related to each indirect measurement.

Starting from the results presented in Table 8.1, it is quite evident that MAHIS methodology in the internal capabilities, interactions, and reorganisation capabilities presents better competence than Gaia and MAS-CommonKADS. Moreover, in the reorganisation capability perspective, the difference is very pronounced because MAHIS supports platform-based open systems. MAHIS has improved the

capability of MAS-CommonKADS in pro-activeness and multiple control. However, the model's dependence of MAHIS has been decreased with the increase of the model complexity of MAHIS.

Table 8.1 Evaluation results of MAHIS

Attributes	Gaia	MC	MAHIS	Attributes	Gaia	Mc	MAHIS
A	**7.83**	**8.33**	**9.58**	C.1.1	0	0	10
A.1	10	10	10	C.1.2	0	0	10
A.2	10	10	10	C.1.3	0	0	10
A.3	3	5	10	C.2	0	0	10
A.4	8.33	8.33	8.33	C.3	0	0	10
A.4.1	5	5	5	C.4	0	0	10
A.4.2	10	10	10	C.5	0	0	7.67
A.4.3	10	10	10	C.5.1	0	0	10
B	**4.7**	**7**	**8**	C.5.2	0	0	3
B.1	8.3	10	10	C.5.3	0	0	10
B.1.1	8	10	10	**D**	**7.5**	**9.5**	**9.46**
B.1.2	10	10	10	D.1	10	8.2	7.85
B.1.2.1	10	10	10	D.1.1	10	10	10
B.1.2.2	10	10	10	D.1.2	10	6.4	5.7
B.1.3	10	10	10	D.2	10	10	10
B.1.4	5	10	10	D.2.1	10	10	10
B.2	5	5	5	D.2.2	10	10	10
B.3	5	5	10	D.3	0	10	10
B.4	5	5	5	D.4	10	10	10
B.5	0	10	10	D.4.1	10	10	10
C	**0**	**1.33**	**9.53**	D.4.2	10	10	10
C.1	0	6.67	10	**Total**	**20.0**	**26.2**	**36.6**

8.4 Suitability Analysis of MAHIS

From the discussion in Section 3.2, it is known that there are four distinct characteristics of agent-based HIS. These characteristics have been covered by the

reorganisation model of MAHIS. In this section, the suitability of MAHIS for constructing agent-based HIS is discussed by analysing the characteristics of HIS and the evaluation results.

- ✓ ***HIS consists of a number of inter-related subsystems organised in a hierarchical fashion***

MAHIS can analyse and design agent applications based on the agent group perspective. There are four tasks in reorganisation model of MAHIS: agent categorising rules, agent grouping rules, agent VO rules, and dynamics rules as discussed in Chapter 4. The agent categorising rules define the hierarchical structure of the system by categorising agents in different level. The agent grouping rules have two aspects of subtasks. One is to define the primitive member for each category. Another one is to define the subsystems each of which can be regarded as an organisation in accordance with the organisation model of MAHIS. The organisation model attempts to describe the real world organisations in context analysis phase. The reorganisation model describes the organisations in concept analysis phase. That is, the output of the organisation model is one of the inputs of the reorganisation model. The interactions between those organisations (agent groups) are defined in the dynamics rules of the reorganisation model.

The subsystem characteristic has been evaluated in the enriched evaluation framework. The social ability (B.1), subsystem interaction (B.5), hierarchical structure (C.1), and application-based reorganisation (C.4) attributes associate the subsystem characteristic of HIS. The mapping results of these attributes are all 10 as shown in Table 8.1. The inter-related subsystems organised in a hierarchical fashion have been definitely covered by MAHIS.

- ✓ ***The choice of the primitive components in each hierarchical level is arbitrary and is defined according to the needs of the observers***

As we have discussed above, one of the subtasks of the agent grouping rules is to define the primitive members for each category. In fact, this definition is flexible. The initial definition of the primitive members for each category is just for the dynamic platform development. After finishing the platform development, the primitive members can be changed according to the needs of the observers. Those

changes do not affect the structure and mechanism of the platform. The changing characteristic of the primitive members for each category has been evaluated in the enriched evaluation framework. The agent as a level member (C.2) attribute associates the primitive member changing characteristic of HIS. The mapping result of this attribute is 10 as shown in Table 8.1. The primitive member changing characteristic has been definitely covered by MAHIS.

- ✓ ***The primitive components in each hierarchical level may be dynamic at unpredictable time***

In the MAHIS methodology, one of the goals of the reorganisation model and design model is to develop dynamic platform. The platform allows the addition and removal of group-based agents at unpredictable time. The description of this ability can be conducted by the dynamics rules in the reorganisation model. The further support of this ability is the agent reusability. The agents in each agent group can be organised before the agent group is added. The agents in an agent group may simultaneously belong to different agent groups, so agents in the system can be reorganised at run-time. The dynamic agents in each level (C.3) attribute associates the dynamic characteristic of HIS. The mapping result of this attribute is 10 as shown in Table 8.1. The dynamic characteristic has been definitely covered by MAHIS.

- ✓ ***The interactions between primitive components may occur at unpredictable times and for unpredictable reasons***

In MAHIS, the coordination model and reorganisation model cope with the interactions between agents. The coordination model deals with the static feature of the interactions between agents. The feature includes the communication channels, message forms, interactions, protocols, etc. The coordination mechanism in the reorganisation model deals with the dynamic interaction problems. The coordination mechanism defines the interaction rules rather than the interactions themselves. Agents interact with others by following those interaction rules. Those interactions may not be known at design time. The interaction with others agents (B.1.2), subsystem interaction (B.5), and coordination mechanism (C.1.3) attributes associate the interaction characteristic of HIS. The mapping results of these

attributes are all 10 as shown in Table 8.1. The interaction characteristic has been definitely covered by MAHIS.

8.5 Summary

This chapter has clarified that the MAHIS methodology is indeed suitable for constructing agent-based HIS as well as the open systems with the hierarchical structure. The comparison results have shown that MAHIS (the total mapping scale 36.6) not only is stronger than Gaia (the total mapping scale 20) and MAS-CommonKADS (the total mapping scale 26.2) in the dynamic ability (*reorganisation attributes*), but also is better in the agent and interaction construction abilities. The *reorganisation attributes* have been proposed according to the characteristics of HIS.

MAHIS has been qualitatively analysed in the following aspects:

- HIS consists of a number of inter-related subsystems organised in a hierarchical fashion;
- The choice of the primitive components in each hierarchical level is arbitrary and is defined according to the needs of the observers;
- The primitive components in each hierarchical level may be dynamic at unpredictable time;
- The interactions between primitive components may occur at unpredictable times and for unpredictable reasons.

The evaluation results have indicated that MAHIS is preferable over Gaia, MAS-CommonKADS, and other methodologies, especially in the construction of HIS as well as the construction of *open systems* with hierarchical structure.

Chapter 9

9 Conclusions and Future Work

In this thesis a methodology for constructing agent-based HIS (called MAHIS) has been proposed based on MAS-CommonKADS. MAS-CommonKADS extended the models defined in CommonKADS (Schreiber, Wielinga et al. 1994, Schreiber, Akkermans et al. 1999), adding techniques from object-oriented methodologies and from protocol engineering to describe the agent protocols. In this research, MAS-CommonKADS is extended again for bridging the gap between it and the agent-based HIS construction. At the same time, the redundant contents of MAS-CommonKADS are cut out because they are not suitable for constructing HIS, or they conflict with the extended parts. MAHIS focuses on the construction of agent-based HIS as well as the *open systems* with hierarchical structure. The development of PAHIS by following MAHIS showed that MAHIS is available and competent in the construction of dynamic platform. We have verified MAHIS in the HIS construction by two case studies: "financial investment planning system" and "petroleum reservoir characterisation system". At last, MAHIS has been evaluated with the enriched *attributes tree* which is known as a qualitative analysis followed by a quantitative rating. The final step is to summarise the arguments presented in the thesis and reflect on them. Section 9.1 discusses the conclusions obtained and possible future work is outlined in Section 9.2.

9.1 Conclusions

Hybrids are essential for complex problem solving because each intelligent technique has particular strengths and limitations and cannot be successfully applied to every type of problem (Zhang 2001). Meanwhile, HIS involves a large number of parts or components that may add to or remove from the systems dynamically. So the design and development of these systems are difficult. Existing software development techniques cannot manage these components and complex interactions efficiently as these interactions may occur at unpredictable times, for unpredictable reasons, and between unpredictable components (Zhang and Zhang 2004).

Agent techniques represent an exciting new means of analysing, designing and building complex software systems. Agent perspective is suitable for modelling, designing, and constructing HIS (Jennings 2001, Zhang and Zhang 2004). However, it is not large to the number of deployed commercial agent-based hybrid intelligent applications. One of the reasons for this is the lack of practical methodologies for agent-based HIS development.

In this thesis, we have followed the following strategy to construct MAHIS which guides the designers to develop their own agent-based HIS as well as any open systems with hierarchical structure.

Firstly, clarify the gap between the existing well-known agent-oriented methodologies and agent-based HIS construction by means of evaluation of these methodologies based on the characteristics of HIS. We have employed and enriched the evaluation framework proposed by Cernuzzi and Rossi (Cernuzzi and Rossi 2002) for comparing and evaluating six well-known agent-oriented methodologies (Gaia, MAS-CommonKADS, MaSE, ODAC, Prometheus, and Tropos). We have enriched the framework with *reorganisation attributes* which are related to the distinct characteristics of HIS. The evaluation results have shown that the current existing agent-oriented methodologies are deficient in HIS construction and MAS-CommonKADS is better than other methodologies.

Secondly, propose MAHIS by extending MAS-CommonKADS for bridging the gap between MAS-CommonKADS and the agent-based HIS construction. MAHIS includes eight models: *Hybrid Integration Strategy Identification Model, Organisation Model*, *Task Model*, *Agent Model*, *Expertise Model*, *Coordination Model*, *Reorganisation Model*, and *Design Model*. Those models are divided into thee levels in accordance with the three process stages of MAHIS: *conceptualisation*, *analysis*, and *design*. Both the *Hybrid Strategy Identification Model* and *Reorganisation Model* are new developed models rather than MAS-CommonKADS. At the same time, some other models have been improved accordingly. The *Reorganisation Model* is the key model to support HIS and *open system*s with hierarchical structure. It consists of *category role*, *group roles*, *virtual organisation role*, and *dynamics rules*. This model describes the hierarchical, dynamic, reusable, and unpredictable characteristics of HIS with *virtual organisation*, *categor*y, and *group* perspectives.

Thirdly, verify MAHIS by case studies. We have successfully developed a dynamic platform for agent-based HIS (PAHIS) and two case studies by following MAHIS. PAHIS with middle agents has been employed in those two systems. Multiple middle agents not only dynamically organise themselves with ring architectural model, but also dynamically manage the registration and cancellation of agent groups.

Finally, evaluate MAHIS with the enriched evaluation framework described in the first stage. We have compared MAHIS with Gaia and MAS-CommonKADS. The evaluation results have indicated that MAHIS is preferable over other methodologies, especially in the construction of HIS.

MAHIS has three distinct characteristics which are not covered by other agent-oriented methodologies. Firstly, MAHIS is suitable for constructing agent-based HIS as well as any *open systems* with hierarchical structure. Secondly, MAHIS supports the construction of agent-based systems with the ability of agent reorganisation. Finally, dynamic platform development is taken into account from the methodology point of view. The platform can dynamically organise all agents in a system.

The main contributions in this thesis include the following five aspects:

1) The gap between six methodologies and agent-based HIS construction has been clarified by evaluating these methodologies with the enriched evaluation framework. The following concrete achievements have been obtained: (a) The characteristics of HIS have been extracted after modelling the hybrid techniques and hybrid strategies; (b) The attributes of the evaluation framework have been enriched in accordance with the hierarchical, dynamic, reusable, and unpredictable characteristics of HIS; (c) Gaia, MAS-CommonKADS, MaSE, ODAC, Prometheus, and Tropos have been ranked based on the attributes associated each hybrid strategy.
2) MAHIS methodology has been proposed by extending MAS-CommonKADS. MAHIS consists of eight models which are divided into three levels: conceptualisation level, analysis level, and design level. Both the *Hybrid Strategy Identification Model* and *Reorganisation Model* are newly developed models rather than from MAS-CommonKADS. At the same time, some other models have been improved accordingly. The *Reorganisation Model* is the key model to support HIS and *open system*s with hierarchical structure. This model describes the hierarchical, dynamic, reusable, and unpredictable characteristics of HIS with *virtual organisation*, *category*, and *group* perspectives. Moreover, a hybrid system development life cycle (HSDLC) followed by MAHIS has been presented.
3) A dynamic platform PAHIS has been developed. PAHIS not only has verified the capability of MAHIS in dynamic platform construction, but also can be used as the infrastructure of agent-based HIS. PAHIS supports the dynamic addition and removal of group-based agents at run-time. A self-organising ring-based architectural model has been proposed to organise the middle agents in PAHIS. Moreover, the ring-based architectural model has been evaluated from the points of view of complexity, efficiency, extendibility, and availability.
4) Two successful case studies: "financial investment planning system" and "petroleum reservoir characterisation system" have been developed for verifying MAHIS in HIS construction. The former has verified MAHIS in constructing

agent-based HIS with tight-coupling hybrid strategy and has been working in laboratory environment. The latter has verified MAHIS in constructing agent-based HIS with loose-coupling hybrid strategy and has been applied in industry. PAHIS has been employed in these two systems.

5) The evaluation of MAHIS with the enriched evaluation framework has been conducted by comparing it with Gaia and MAS-CommonKADS. The evaluation results have indicated that MAHIS is preferable over Gaia and MAS-CommonKADS, especially in the construction of agent-based HIS.

9.2 Future Work

There are two major technical impediments to the widespread adoption of agent technology in the moment (Zhang 2001). The first one is the lack of a systematic methodology enabling designers to clearly specify and structure their applications as multi-agent system. The second one is the lack of widely available industrial-strength multi-agent system toolkits. In response to the first impediment, we have proposed MAHIS which is suitable for the analysis and design of agent-based HIS. However, the development of an appropriate methodology is a long and difficult task. The development of an agent-oriented methodology is even more difficult (Gervais 2003). Further work is needed to improve MAHIS, by:

- Proposing the implementation and maintenance process stages of MAHIS. The hybrid system development life cycle (HSDLC) proposed in this thesis includes five different process stages: conceptualisation, analysis, design, implementation, and maintenance. However, at the moment, MAHIS only involves three of the five process stages: conceptualisation, analysis, and design. The implementation and maintenance process stages need to be formalised and added to MAHIS according to the developed case studies and forthcoming applications and systems.
- Providing suitable normal notations for uniformly expressing the expected outputs of the analysis and design phases of MAHIS. We expect standard

notations, for example, UML Version 2.0 (Huget 2003), to be rapidly adapted to the needs of agent-based software engineering, as well as the methodology: MAHIS.

- Improving MAHIS by using novel ideas of some methodologies, for example, team-oriented methodologies.
- Documenting MAHIS to make it completely understandable.

In response to the second impediment, we have developed a dynamic platform PAHIS as the infrastructure of agent-based HIS. For facilitating the development of agent-based HIS quickly, an industrial-strength toolkit is needed. This toolkit will be based on the proposed MAHIS and developed PAHIS. When developing the toolkit, graphical tools for supporting all phases of the agent construction process are provided like the integrated agent development toolkit: AgentBuilder (AgentBuilder 2000). AgentBuilder provides a comprehensive set of tools for programming software agents.

In addition to these two main aspects, more case studies using MAHIS should be done for further verifying and enriching MAHIS.

Bibliography

AgentBuilder (2000) User's Guide V1.3, Reticular Systems, Inc., San Diego, pp.1-329.

Allsopp, D. J., Harrison, A. and Sheppard, C. (2002) A database architecture for reusable CommonKADS agent specification components, *Knowledge-Based Systems,* **15** (2002), 275-283.

Ao, S. I. (2003) Hybrid intelligent system for pricing the indices of dual-listing stock markets, In Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology, 2003, Halifax, Canada, pp.495-498.

Averbukh, A. B. (1999) Hybrid intelligent architecture for real time processing, In Proceedings of the International Joint Conference on Neural Networks (IJCNN '99), Washington, DC USA, Vol.6, pp.4107-4110.

Balducelli, C. and D'Esposito, C. (2000) Genetic agents in an EDSS system to optimize resources management and risk object evacuation, *Safety Science,* **35** (1-3), 59-73.

Barhen, J. (2002) Reduction of uncertainties in neural network prediction of oil well logs, In Proceedings of 2002 International Joint Conference on Neural Networks (IJCNN '02), Honolulu, HI USA, Vol.1, pp.902-907.

Bauer, B., Muller, J. P. and Odell, J. J. (2001) Agent UML: a formalism for specifying multiagent interaction, In *Agent-Oriented Software Engineering*,

(Eds, Ciancarini, P. and Wooldridge, M.) Springer, Berlin, Germany, pp.91-103.

Bento, J. and Feijo, B. (1997) An agent-based paradigm for building intelligent CAD systems, *Artificial Intelligence in Engineering,* **11** (3), 231-244.

Bienvenido, J. F. and Flores-Parra, I. M. (2003) STEM: a methodology for the development of multiagent design tools using a general knowledge model of configurational design, In Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2003, Boston, USA, pp.602 - 607.

Booch, G. (1994) *Object-oriented analysis and design with applications,* Addison Wesley.

Bourne, R. A., Shoop, K. and Jennings, N. R. (2001) Dynamic evaluation of coordination mechanisms for autonomous agents, In Proceedings of 10th Portuguese Conference on Artificial Intelligence, Porto, Portugal, pp.155-168.

Brazier, F. M. T., Dunin-keplicz, B. M., Jennings, N. R. and Treur, J. (1997) DESIRE: modelling multi-agent systems in a compositional formal framework, *International Journal of Cooperative Information Systems,* **6** (1), 67-94.

Bresciani, P. and Giorgini, P. (2002) The TROPOS analysis process as graph transformation system, In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA, pp.1-12.

Brussel, H. V., Wyns, J., Valckenaers, P., Bongaerts, L. and Peeters, P. (1998) Reference architecture for holonic manufacturing systems: PROSA, *Computers in Industry,* **37** (3), 255-274.

Cabri, G., Leonardi, L. and Zamboneli, F. (2002) Modeling role-based interactions for agents, In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA, pp.13-20.

Cao, L., Li, C., Zhang, C. and Dai, R. (2003) Open giant intelligent information systems and its agent-oriented analysis and design, In Proceedings of 2003

International Conference on Software Engineering Research and Practice (SERP'03), Las Vegas, USA, Vol.2, pp.816-822.

Centeno-Gonzalez, J., Velasco, J. R. and Iglesias, C. A. (1999) An agent-based operational model for hybrid connectionist-symbolic learning, In Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, Alicante, Spain, Vol.2, pp.50-57.

Cernuzzi, L. and Rossi, G. (2002) On the evaluation of agent oriented methodologies, In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA, pp.21-30.

Cuesta, P., Gomez, A., Gonzalez, J. C. and Rodriguez, F. (2004) A framework for evaluation of agent oriented methodologies, University of Vigo, http://ma.ei.uvigo.es/aose/evaluation.php, pp.1-10.

Dam, K. H. and Winikoff, M. (2003) Comparing agent-oriented methodologies, In *Agent-Oriented Information Systems*, (Eds, Giorgini, P., Henderson-Sellers, B. and Winikoff, M.) LNCS 3030, Springer, Berlin, Germany, pp.78-93.

Decker, K., Sycara, K. and Williamson, M. (1997) Middle-agents for the Internet, In Proceedings of the 15th International Joint Conference on Artificial Intelligence, Nagoya, Japan, Vol.1, pp.578-583.

del Aguila, I. M., Canadas, J., Tunez, S. and Palma-Mendez, J. T. (2003) the Integration of development methodologies for the building of knowledge intensive multiagent systems, In Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2003, Boston, USA, pp.203-208.

Delgado, M. and Gomez-Skarmeta, A. F. (1999) A multi-agent architecture for fuzzy modeling, *International Journal of Intelligent Systems,* **14** (1999), 305-329.

Deloach, S. A., Wood, M. F. and Sparkman, C. H. (2001) Multiagent systems engineering, *International Journal of Software Engineering and Knowledge Engineering,* **11** (3), 231-258.

Feijo, B. and Bento, J. (1998) A logic-based environment for reactive agents in intelligent CAD systems, *Advances in Engineering Software,* **29** (10), 825-832.

Ferber, J., Gutknecht, O. and Michel, F. (2003) From agents to organization: an organizational view of multi-agent systems, In *Agent-Oriented Software Engineering IV*, (Eds, Giorgini, P., Muller, J. P. and Odell, J.) LNCS 2935, Springer, Berlin, Germany, pp.214-230.

Finin, T., Labrou, Y. and Mayfield, J. (1997) KQML as an agent communication language, In *Software Agents*, (Ed, Brashaw, J. M.) AAAI Press/The MIT Press, pp.291-316.

Fisher, M. and Wooldridge, M. (1997) On the formal specification and verification of multi-agent systems, *International Journal of Cooperative Information Systems,* **6** (1), 37-65.

Fregene, K., Kennedy, D. and Wang, D. (2003) Multi-vehicle pursuit-evasion: an agent-based framework, In Proceedings of IEEE International Conference on Robotics and Automation (ICRA '03), Vol.2, pp.1050-4729.

Fregene, K., Madhavan, R. and Kennedy, D. (2004) Coordinated control of multiple terrain mapping UGVs, In Proceedings of IEEE International Conference on Robotics and Automation (ICRA '04), Vol.4, pp.1050-4729.

Georgeff, M., Pell, B., Pollack, M., Tambe, M. and Wooldridge, M. (1999) The belief-desire-intention model of agency, In *Intelligent Agents V*, (Eds, Muller, J. P., Singh, M. and Rao, A.) LNCS 1365, Springer, Berlin, Germany.

Gervais, M.-P. (2003) ODAC: an agent-oriented methodology based on ODP, *Autonomous Agents and Multi-Agent Systems,* **7** (3), 199-228.

Giunchiglia, F., Mylopoulos, J. and Perini, A. (2003) The Tropos software development methodology: processes, models, and diagrams, In *Agent-oriented Software Engineering III*, (Eds, Giunchiglia, F., Odell, J. and WeiB, G.) LNCS 2585, Springer, Berlin, Germany, pp.162-173.

Goldman, C. V. and Rosenschein, J. S. (2002) Evolutionary patterns of agent organizations, *IEEE Transactions on Systems, Man, and Cybernetics,* **32** (1), 135-148.

Goonatilake, S. and Khebbal, S. (1995) *Intelligent hybrid systems,* John Wiley & Sons Ltd, Chichester, England.

Graham, I., Henderson-Sellers, B. and Younessi, H. (1997) *The OPEN process specification,* Addison-Wesley.

Graham, J. R., Decker, K. S. and Mersic, M. (2003) DECAF - a flexible multi-agent system architecture, *Autonomous Agents and Multi-Agent Systems,* **7** (2003), 7-27.

Hanachi, C. and Sibertin-Blanc, C. (2004) Protocol moderators as active middle-agents in multi-agent systems, *Autonomous Agents and Multi-Agent Systems,* **8** (2), 131-164.

Hu, H. and Brady, M. (1996) A parallel processing architecture for sensor-based control of intelligent mobile robots, *Robotics and Autonomous Systems,* **17** (4), 235-257.

Huang, Y., Wong, P. M. and Gedeon, T. D. (1996) An improved fuzzy neural network for permeability estimation from wireline logs in a petroleum reservoir, In Proceedings of TENCON '96, Perth, WA Australia, Vol.2, pp.912-917.

Huget, M.-P. (2002) Nemo: an agent-oriented software engineering methodology, In Proceedings of the OOPSL 2002 Workshop on agent-oriented methodologies, Seattle, USA, pp.43-53.

Huget, M.-P. (2003) FIPA Modeling: Interaction Diagrams, Foundation for Intelligent Physical Agents, Geneva, pp.1-45.

Huynh, T. D., Jennings, N. R. and Shadbolt, N. (2004) FIRE: An integrated trust and reputation model for open multi-agent systems, In Proceedings of 16th European Conference on Artificial Intelligence, Valencia, Spain, pp.18-22.

Iglesias, C. A., Centeno-Gonzalez, J. and Velasco, J. R. (1995) MIX: a general purpose multiagent architecture, In Proceedings of ATAL, Montreal, Canada, pp.251-266.

Iglesias, C. A., Centeno-Gonzalez, J. and Velasco, J. R. (1998) A fuzzy-neural multiagent system for optimisation of a roll-mill application, In Proceedings of 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Castellon, Spain, pp.596-605.

Iglesias, C. A., Garijo, M. and Centeno-Gonzalez, J. (1998) A survey of agent-oriented methodologies, In Proceedings of 5th International Workshop, ATAL '98, Paris, France, pp.317-330.

Iglesias, C. A., Garijo, M., Centeno-Gonzalez, J. and Velasco, J. R. (1997) Analysis and design of multiagent systems using MAS-CommonKADS, In Proceedings of 4th International Workshop, ATAL '97, Providence, Rhode Island, USA, pp.313-327.

Iglesias, C. A., Garijo, M., Gonzalez, J. C. and Velasco, J. R. (1996) A methodological proposal for multiagent systems development extending CommonKADS, In Proceedings of 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, Vol.1, pp.25-1/17.

Jacobsen, H. A. (1998) A generic architecture for hybrid intelligent systems, In Proceedings of the 1998 IEEE International Conference on Fuzzy Systems, Anchorage, AK USA, Vol.1, pp.709-714.

Jain, L. C. and Jain, R. K. (Eds.) (1997) *Hybrid Intelligent Engineering Systems,* World Scientific Publishing Co. Pte. Ltd., Singapore.

Jennings, N. R. (2000) On agent-based software engineering, *Artificial Intelligence,* **117** (2), 145-189.

Jennings, N. R. (2001) An agent-based approach for building complex software systems, *Communications of the ACM,* **44** (4), 35-41.

Jennings, N. R., Sycara, K. and Wooldridge, M. (1998) A roadmap of agent research and development, *Autonomous Agents and Multi-Agent Systems,* **1** (1998), 7-38.

Jennings, N. R. and Wooldridge, M. (1996) Software agents, *IEE Review,* 17-20.

Jouvin, D. and Hassas, S. (2003) Dynamic multi-agent architecture using conversational role delegation, In *Agent-Oriented Software Engineering IV*,

(Eds, Giorgini, P., Muller, J. P. and Odell, J.) LNCS 2935, Springer, Berlin, pp.185-200.

Juan, T., Stering, L., Martelli, M. and Mascardi, V. (2003) Customizing AOSE methodologies by reusing AOSE features, In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia, pp.113-120.

Kaminka, G. A., Frank, I., Arai, K. and Tanaka-Ishii, K. (2003) Performance competitions as research infrastructure: large scale comparative studies of multi-agent teams, *Autonomous Agents and Multi-Agent Systems,* **7** (2003), 121-144.

Khosla, R. and Dillon, T. (1997) *Engineering intelligent hybrid multi-agent systems,* Kluwer Academic Publishers, Boston, USA.

Kim, B.-I., Heragu, S. S., Graves, R. J. and St Onge, A. (2003) A hybrid scheduling and control system architecture for warehouse management, *IEEE Transactions on Robotics and Automation,* **19** (6), 991-1001.

Kingston, J. K. C. (1998) Designing knowledge based systems: the CommonKADS design model, *Knowledge-BASED Systems,* **11** (1998), 311-319.

Kinny, D. and Georgeff, M. P. (1997) Modelling and design of multi-agent systems, In Proceedings of ATAL, Budapest, Hungary, pp.1-20.

Kinny, D., Georgeff, M. P. and Rao, A. S. (1996) A methodology and modelling technique for systems of BDI agents, In Proceedings of 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, pp.56-71.

Klein, M. (2003) A knowledge-based methodology for designing reliable multi-agent systems, In *Agent-Oriented Software Engineering IV*, (Eds, Giorgini, P., Muller, J. P. and Odell, J.) LNCS 2935, Springer, Berlin, pp.85-95.

Klostermeyer, A. and Klemm, E. (2003) PABADIS - an agent based flexible manufacturing concept, In Proceedings of IEEE International Conference on Industrial Informatics (INDIN 2003), pp.286-293.

Lahlouhi, A. and Sahnoun, Z. (2002) Multi-agent methodologies' incoherencies, In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA, pp.64-73.

Lauber, J., Steger, C. and Weiss, R. (1999) Autonomous agents for online diagnosis of a safety-critical system based on probabilistic causal reasoning, In Proceedings of the Fourth International Symposium on Autonomous Decentralized Systems, Integration of Heterogeneous Systems, Tokyo Japan, pp.213-219.

Lertpalangsunti, N. and Chan, C. W. (1998) An architecture framework for the construction of hybrid intelligent forecasting systems: application for electricity demand prediction, *Engineering Application of Artificial Intelligence,* **11** (4), 549-565.

Li, C., Cheng, D. and Zhang, C. (2004) A platform to integrate well-log information applications on heterogeneous environments, In Proceedings of the 2nd International Conference on Information Technology and Applications (ICITA2004), Harbin, China, pp.265-270.

Li, C., Liu, L. and Song, Q. (2004) A practical framework for agent-based hybrid intelligent systems, *Asian Journal of Information Technology,* **3** (2), 107-114.

Li, C., Song, Q., Wang, M. and Zhang, C. (2003) SCTR: an approach to digitizing well-logging graph, In Proceedings of the IASTED International Conference on Computer, Graphics and Image (CGIM 2003), Honolulu, USA, pp.285-288.

Li, C., Wang, M. and Yang, L. (2002) An agent-based system for well-logging data manipulation on Intranet, *Daqing Petroleum Institute Journal,* **26** (1), 54-56.

Li, C., Zhang, C. and Cao, L. (2003a) Theoretical evaluation of ring-based architectural model for middle agents in agent-based system, In Proceedings of the 14th International Symposium on Methodologies for Intelligent Systems (ISMIS'03), Maebashi, Japan, pp.603-607.

Li, C., Zhang, C., Chen, Q. and Zhang, Z. (2003b) A scalable and robust framework for agent-based heterogeneous database operation, In Proceedings of the

International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC2003), Vienna, Austria, pp.260-271.

Li, C., Zhang, C. and Wang, M. (2003c) An agent-based curve-digitizing system for well-logging data management, In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2003), Las Vegas, USA, pp.656-660.

Li, C., Zhang, C., Wang, M. and Song, Q. (2003d) An approach to digitizing and managing well-logging graphs with agent-based perspective, In Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT-2003), Halifax, Canada, pp.11-17.

Li, C., Zhang, C. and Zhang, Z. (2002a) An agent-based middleware for uniform operation in a heterogeneous database environment, In Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02), Singapore, Vol.1, pp.385-389.

Li, C., Zhang, C. and Zhang, Z. (2002b) An agent-based intelligent system for information gathering from World Wide Web environment, In Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, China, Vol.4, pp.1852-1857.

Li, C., Zhang, C. and Zhang, Z. (2003e) A ring-based architectural model for middle agents in agent-based system, In Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL2003), Hon Kong, pp.94-98.

Li, C., Zhang, Z. and Zhang, C. (2004) A platform for dynamic organization of agents in agent-based systems, In Proceedings of the 2004 IEEE/WIC International Conference on Intelligent Agent Technology (IAT04), Beijing, China, pp.160-163.

Li, S. (2000) The development of a hybrid intelligent system for developing marketing strategy, *Decision Support Systems,* **27** (2000), 395-409.

Liu, K., Sun, L., Barjis, J. and Dietz, J. L. G. (2003) Modelling dynamic behaviour of business organisations -- extension of DEMO from a semiotic perspective, *Knowledge-Based Systems,* **16** (2), 101-111.

Luo, X., Zhang, C. and Jennings, N. R. (2002) A hybrid model for sharing information between fuzzy, uncertain and default reasoning models in multi-agent systems, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* **10** (4), 401-450.

Mali, A. D. (2003) On the evaluation of agent behaviors, *Artificial Intelligence,* **143** (2003), 1-17.

Marwaha, S., Chen, K. T. and Srinivasan, D. (2002) A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks, In Proceedings of 10th IEEE International Conference on Networks (ICON 2002), pp.311-316.

Medina, M. A., Sanchez, A. and Castellanos, N. (2004) Ontological agents model based on MAS-CommonKADS methodology, In Proceedings of 14th International Conference on Electronics, Communications and Computers, 2004 (CONIELECOMP 2004). Veracruz, MEXICO, pp.260 - 263.

Medsker, L. R. (1995) *Hybrid intelligent systems,* Kluwer Academic Publishers, Boston, USA.

Medsker, L. R. and Bailey, D. L. (1992) Models and guidelines for integrating expert system and neural networks, In *Hybrid Architectures for Intelligent Systems*, (Eds, Kandel, A. and Langholz, G.) CRC Press, Boca Raton, pp.154-171.

Monostori, L. (2003) AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing, *Engineering Applications of Artificial Intelligence,* **16** (4), 277-291.

Mouratidis, H., Giorgini, P., Manson, G. and Philp, I. (2002) A natural extension of Tropos methodology for modelling security, In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA, pp.74-85.

Mrhailaf, R. and Sahraoui, A. (1993) DFD extended methods for specifying hybrid systems, In Proceedings of the International Conference on Systems, Man and Cybernetics, Le Touquet France, pp.681-686.

Nair, R., Tambe, M., Marsella, S. and Raines, T. (2004) Automated assistants for analyzing team behaviors, *Autonomous Agents and Multi-Agent Systems,* **8** (1), 69-111.

Norman, T. J., Preece, A., Chalmers, S., Jennings, N. R., Luck, M., Dang, V. D., Nguyen, T. D., Deora, V., Shao, J., Gray, A. and Fiddian, N. (2003) CONOISE: Agent-based formation of virtual organisations, In Proceedings of 23rd SGAI Int. Conf. on Innovative Techniques and Applications of AI, Cambridge, UK, pp.353-366.

Odell, J. J., Parunak, H. V. D. and Bauer, B. (2000) Extending UML for agents, In Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, Austin, TX, pp.3-17.

Odell, J. J., Parunak, H. V. D. and Bauer, B. (2001) Representing agent interaction protocols in UML, In *Agent-Oriented Software Engineering*, (Eds, Ciancarini, P. and Wooldridge, M.) Springer, Berlin, Germany, pp.121-140.

Odell, J. J., Parunak, H. V. D., Brueckner, S. and Sauter, J. (2003a) Temporal aspects of dynamic role assignment, In *Agent-Oriented Software Engineering IV*, (Eds, Giorgini, P., Muller, J. P. and Odell, J.) LNCS 2935, Springer, Berlin, Germany, pp.201-213.

Odell, J. J., Parunak, H. V. D. and Fleischer, M. (2003b) The role of roles in designing effective agent Organizations, In *Software Engineering for Large-Scale Multi-Agent Systems*, (Eds, Garcia, A., Lucena, C., Zambonelli, F., Omicini, A. and Castro, J.) LNCS 2603, Springer, Berlin, pp.27-38.

Odell, J. J., Parunak, H. V. D., Fleischer, M. and Breuckner, S. (2002) Modeling agents and their environment, In *Agent-Oriented Software Engineering (AOSE) III*, (Eds, Giunchiglia, F., Odell, J. J. and Weiss, G.) LNCS 2585, Springer, Berlin, pp.16-31.

O'Malley, S. A. and Deloach, S. A. (2001) Determining when to use an agent-oriented software engineering paradigm, In Proceedings of the Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada, pp.1-18.

Padgham, L. and Winikoff, M. (2002) Prometheus: a pragmatic methodology for engineering intelligent agents, In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, USA, pp.97-108.

Padgham, L. and Winikoff, M. (2003) Prometheus: a methodology for developing intelligent agents, In *Agent-Oriented Software Engineering III*, Vol. 2585 (Eds, Giunchiglia, F., Odell, J. and WeiB, G.) LNCS 2585, Springer, Berlin, Germany, pp.174-185.

Parunak, H. V. D. and Odell, J. J. (2002) Representing social structures in UML, In *Agent-Oriented Software Engineering II*, (Eds, Wooldridge, M., Weiss, G. and Ciancarini, P.) Springer, Berlin, Germany, pp.1-16.

Post, W., Wielinga, B., De Hoog, R. and Schreiber, G. (1997) Organizational modeling in CommonKADS: the emergency medical service, *IEEE Expert,* **11** (2), 74-76.

Rudolph, E., Graubman, P. and Grabowski, J. (1996) Tutorial on Message Sequence Charts, *Computer Networks and ISDN Systems,* **28** (12), 1629-1641.

Russell, S. and Norvig, P. (1995) *Artificial Intelligence: a Modern Approach,* Prentice-Hall, Englewood Cliffs, NJ.

Sandberg, J. and de Hoog, R. (1996) Using CommonKADS in "soft" domains, In Proceedings of Mexico-USA Collaboration in Intelligent Systems Technologies (ISAI/IFIS 1996), Cancun, Mexico, pp.262 - 268.

Scherer, A. and Schlageter, G. (1995) A multi-agent approach for the integration of neural networks and expert systems, In *Intelligent Hybrid Systems*, (Eds, Goonatilake, S. and Khebbal, S.) Wiley, pp.153-173.

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. d., Shadbolt, N. and Velde, W. v. d. (1999) *Knowledge engineering and management: the CommonKADS methodology,* MIT Press.

Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H. and Van de Velde, W. (1994) CommonKADS: a comprehensive methodology for KBS development, *IEEE Expert,* **9** (6), 28-37.

Shehory, O. and Sturm, A. (2001) Evaluation of modeling techniques for agent-based systems, In Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Quebec, Canada, pp.624-631.

Sierra, C., Faratin, P. and Jennings, N. R. (1997) A service-oriented negotiation model between autonomous agents, In Proceedings of 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, Ronneby, Sweden, pp.17-35.

Simon, H. (1996) *The sciences of the artificial,* MIT Press.

Sobh, T. M. and Bajcsy, R. (1992) Visual observation under uncertainty as a discrete event process, In Proceedings of 11th IAPR International Conference on Pattern Recognition, The Hague Netherlands, pp.429-432.

Soleng, H. H. (1999) Oil reservoir production forecasting with uncertainty estimation using genetic algorithms, In Proceedings of the 1999 Congress on Evolutionary Computation, Washington, DC USA, Vol.2, pp.1223.

Sommerville, I. (1995) *Software engineering,* Addion-Wesley, Harlow, UK.

Srikanth, V. (1999) Intelligent trading systems: a multi-agent hybrid architecture, In Proceedings of IEEE/IAFE 1999 Conference on Computational Intelligence for Financial Engineering, New York, NY USA, pp.64-73.

Studer, R., Benjamings, V. R. and Fensel, D. (1998) Knowledge engineering: principles and methods, *Data & Knowledge Engineering,* **25** (1998), 161-197.

Sturm, A. and Shehory, O. (2003) A framework for evaluating agent-oriented methodologies, In *Agent-Oriented Information Systems*, (Eds, Giorgini, P., Henderson-Sellers, B. and Winikoff, M.) LNCS 3030, Springer, Berlin, Germany, pp.94-109.

Sudeikat, J., Braubach, L., Pokahr, A. and Lamersdorf, W. (2004) Evaluation of agent-oriented software methodologies ---- examination of the gap between modeling and platform, In Proceedings of AAMAS Agent-Oriented Software Engineering (AOSE) Workshop, New York, USA, pp.33-48.

Tanenbaum, S. (1995) *Distributed operating systems,* Prentice-Hall, Inc., New Jersey, USA.

Turner, K. J. (1993) *Using formal description techniques,* John Wiley and Sons Ltd, Chichester, England.

van Breemen, A. J. N. and de Vries, T. J. A. (2001) Design and implementation of a room thermostat using an agent-based approach, *Control Engineering Practice,* **9** (3), 233-248.

Varga, L., Jennings, N. R. and Cockburn, D. (1994) Integrating intelligent systems into a cooperating community for electricity distribution management, *Expert Systems with Applications,* **7** (4), 563-579.

Velasco, J. R., Gonzalez, J. C., Magdalena, L. and Iglesias, C. A. (1996) Multiagent-based control systems: A hybrid approach to distributed process control, *Control Engineering Practice,* **4** (6), 839-845.

Wong, P. M., Bruce, A. G. and Gedeon, T. D. (2002) Confidence bounds of petrophysical predictions from conventional neural networks, *IEEE Transactions on Geoscience and Remote Sensing,* **40** (6), 1440-1444.

Wong, P. M., Gedeon, T. D. and Taggart, I. J. (1995) An improved technique in porosity prediction: a neural network approach, *IEEE Transactions on Geoscience and Remote Sensing,* **33** (4), 971-980.

Wood, M. F. and Deloach, S. A. (2001) An overview of the multiagent systems engineering methodology, In *Agent-Oriented Software Engineering*, (Eds, Ciancarini, P. and Wooldridge, M.) LNCS 1957, Springer, Berlin, Germany, pp.207-222.

Wooldridge, M. (1997) Agent-based software engineering, *IEE Proc. Software Engineering,* **144** (1), 26-37.

Wooldridge, M. (1998) Agents and software engineering, *AI*IA Notizie,* **XI** (3), 31-37.

Wooldridge, M. (1999) Intelligent agent, In *Multiagent System*, (Ed, Weiss, G.) The MIT Press.

Wooldridge, M. (2001) *An introduction to multiagent systems,* John Wiley & Sons Ltd, Chichester, England.

Wooldridge, M. and Ciancarini, P. (2001) Agent-oriented software engineering: the start of the art, In *Agent-Oriented Software Engineering*, (Eds, Ciancarini, P. and Wooldridge, M.) Springer, Berlin, pp.1-28.

Wooldridge, M. and Jennings, N. R. (1995) Intelligent agents: theory and practice, *The Knowledge Engineering Review,* **10** (2), 115-152.

Wooldridge, M. and Jennings, N. R. (1999) Software engineering with agents: pitfalls and pratfalls, *IEEE Internet Computing,* **3** (3), 20-27.

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

Yan, Q., Mao, X., Shan, L., Qi, Z. and Zhu, H. (2003) Soft gene, role, agent: MABS learns from sociology, In Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), Halifax, Canada, pp.450-453.

Yang, C. C., Yen, J. and Chen, H. (2000) Intelligent internet searching agent based on hybrid simulated annealing, *Decision Support Systems,* **28** (3), 269-277.

Zambonelli, F., Jennings, N. R., Omicini, A. and Wooldridge, M. (2001) Agent-oriented software engineering for Internet applications, In *Coordination of Internet Agents: Models, Technologies and Applications*, (Eds, Omicini, A., Zambonelli, F., Klusch, M. and Tolksdorf, R.) Springer, Berlin, Germany, pp.326-346.

Zambonelli, F., Jennings, N. R. and Wooldridge, M. (2000) Organisational abstractions for the analysis and design of multi-agent systems, In Proceedings of 1st International Workshop on Agent-Oriented Software Engineering, Limerick, Ireland, pp.127-141.

Zambonelli, F., Jennings, N. R. and Wooldridge, M. (2003) Developing Multiagent Systems: The Gaia Methodology, *ACM Trans on Software Engineering and Methodology,* **12** (3), 317-370.

Zha, X. F., Lim, S. Y. E. and Fok, S. C. (1997) Concurrent intelligent design and assembly planning (CIDAP): object-oriented intelligent Petri nets (OOIPNs)

approach, In Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL USA, Vol.4, pp.3930-3935.

Zhang, C., Li, C. and Zhang, Z. (2002) An agent-based framework for petroleum information services from distributed heterogeneous data resources, In Proceedings of the 9th Asia Pacific Software Engineering Conference (APSEC 2002), Gold Coast, Australia, pp.593-602.

Zhang, S.-Z., Li, C.-L. and Lu, Z.-D. (2003) Design and implementation of a hybrid intelligent and mobile agent platform, In Proceedings of 2003 International Conference on Machine Learning and Cybernetics, Xi'an, China, Vol.4, pp.2025-2030.

Zhang, Z. (2001) An agent-based hybrid framework for decision making on complex problems, PhD thesis, School of Information Technology, Deakin University, Geelong, Australia.

Zhang, Z. and Zhang, C. (2000a) Agent-oriented approaches to constructing hybrid intelligent systems, In Proceedings of 7th International Conference on Neural Information Processing, Taejon, Korea, Vol.1, pp.258-263.

Zhang, Z. and Zhang, C. (2000b) Result fusion in multi-agent systems based on OWA operator, In Proceedings of 23rd Australian Computer Science Conference, Canberra, Australia, pp.234-240.

Zhang, Z. and Zhang, C. (2004) *Agent-based hybrid intelligent systems,* LNAI 2938, Springer, Berlin, Germany.