A Framework For Multi-agent Cooperation Applied To Multiple Robots

By

Gangmin Li

Thesis submitted for the degree of

Doctor of Philosophy

in the Faculty of Technology

Department of Telematics,

Faculty of Technology, The Open University,

Milton Keynes, MK7 6AA, United Kingdom

September 1999

Abstract

Cooperation between multiple robots is being used to tackle increasingly complex tasks. The amount of knowledge needed to build multi-robot systems is becoming unwieldy and it is difficult to engineer a closed system. One paradigm for overcoming this barrier is to organise the basic components of the systems in a manner which allows the sharing of performance, responsibilities, and resources. Until recently, research in multi-agent systems has been based on *ad hoc* models of action and interaction. The notion of intention is beginning to emerge as a prime candidate upon which a sound theory could be based.

This research, motivated by questioning the suitability of the intentional theory, implements two of the most influential frameworks into a multi-robot domain to test their applicability and to identify any possible inadequacies. To refine the existing frameworks, this research proposes a new framework which is inspired by organisation theory and economic team theory. Shifting Matrix Management (SMM) divides agents' actions during the cooperation process into six stages, namely goal selection, act selection, team formation, plan formation and shifting. The mental states and the positions of agents in such a process are viewed as continually shifting in a matrix structure. To support the framework, a quantitative decision theory is developed. The theory reveals that an agent's mental state on selecting an action is defined by three quantitative functions: a probability density function, an outcome function, and a utility function. The framework has been formalised into a general and abstract model.

The SMM model was implemented in the control of cooperation between multiple robots and a number of controlled experiments were performed to evaluate the model. The results indicate the effectiveness of the model. It is believed that the effectiveness of the SMM model may be generalisable.

Acknowledgements

Many thanks to both my supervisors Dr. Adrian Hopgood and Dr. Martin Weller for their invaluable guidance and continued assistance throughout the research process. They are not only the supervisors but also friends. They have not only guided me to obtain knowledge from the research area but also from life, the scientific research attitude, an earnest manner towards problems and a practical and realistic way of doing things. This knowledge has not only made my stay at The Open University more fruitful, interesting and enjoyable but also will benefit my whole life.

I also want to thank various colleagues and friends for their suggestions, comments and criticisms. Here I include Prof. John Monk, Chris Dillon, Nicky Moss, David Reed, David Chapman, and Ahmad Kharaz. I also want to thank Anthony Lucas-Smith for his help on my English. I am grateful for the support of the Technical Support Group and the departmental secretary.

My warmest thanks go to my wife Hu Ling and my son Botong Li for their understanding, support and company. My thanks also go to my parents in China for their moral support and encouragement.

I would like to thank The Open University for providing my research grant. Clearly without it, I would not have stayed here to do the research.

Contents

Chapter 1

Introduction

"There are three robots, and let them make a cup of tea. ..." Said my supervisor.

"That is easy." I thought. I can quickly write a tea-making program, and let them execute my instructions. One boils water, another brings a cup, and the third one fetches a tea bag, then puts it into the empty cup, finally the first one pours boiled water into the cup. The program appears in my mind already.

"... ...There is no central controller in the system. The available number of robots is even changeable. And they are all autonomous; different in functionality, control mechanism. They can be situated in the different areas, one of them may not even work properly at the present moment ... " my supervisor was continuing,

"..." I was defeated.

-- Gangmin Li's diary, 21 October 1995

This chapter provides the background, aim, objectives, motivation and context for this research. Starting with the description of the problem domain and its isomorphic problem family, Distributed Artificial Intelligence (DAI) approach and its supporting theory are introduced. The initial research motivation is grounded on analysis of the deficiencies of current theory. The main research context and original contribution of this thesis are then given by briefly reviewing the two major subfields of DAI, distributed problem solving (DPS) and cooperative problem solving (CPS). Finally, the outline of the thesis is provided.

1.1 Research Background, Aim and Objectives

Research into the cooperation of multiple robots is widely recognised as a step towards high performance and flexibility in industrial automation [Cardarelli 94(1), Koivo and Bekey 88, Paljug and Yun 93]. Firstly, cooperating robots can perform tasks that are either difficult or impossible for a single robot. Secondly, distributed task performance amongst multiple robots can result in reliable performance and graceful degradation. Thirdly, well organised multiple robots working in parallel and cooperatively can achieve high efficiency. Finally, multiple robots sharing resources such as space, tools, feeder, vision system and so forth, will reduce the cost of the system's scale-up without a drop in productivity [Shin 85].

In an industrial environment, there are normally a number of robots pre-existing and performing various tasks. There are tasks that can not be performed by robots working in isolation or which need to be performed in a cooperative way to increase the efficiency. The aim of this research is to develop a framework for multiple robots' cooperation.

To achieve the above research aim, the specifications of the problem need to be studied. A system in which cooperation between multiple robots is taking place can be called a multi-robot system. Firstly, a robot in a multi-robot system is a stand-alone entity that can have a different functionality, control mechanism and servicing history. Secondly, the tasks, which need to be performed by the multi-robot systems, are varied in both complexity and scale. Thirdly, new robots may continuously join the system to replace older ones. The number of robots engaged in a particular task's performance in the system is variable depending on the tasks and the robot's availability. Lastly, modification to existing robots in the system in order to achieve cooperation is required

to be kept to a minimum. This means that the cost of implementing the new cooperation framework in both engineering and financial senses should be minimal.

With the aim of the research and the specifications of the problem stated above, the following four objectives of the research have been identified:

1. Develop an applicable framework for multiple robots' cooperation.
2. Develop a theory, which can assess cooperative actions.
3. Develop a model to clearly map from theory to application.
4. Demonstrate the practical effectiveness of the framework.

1.2 Motivation

The initial motivation of this research is not only to develop a framework for multi-robot system, rather it deals with a broader area which is classified as DAI. This is because cooperation of multiple robots should not be viewed as a stand-alone problem in the author's opinion, but as part of a larger isomorphic problem family. The problems in this isomorphic problem family range from the mainstream of computing science [Newell 82, Jennings and Campos 97], software engineering [Shoham 93, Lander 95], information systems [Papazoglou 92] to real world applications like air traffic control [Steeb 88], manufacturing automation [Parunak 96], computer supported cooperative work [Greif 88, Maes 94] and electronic commerce [Chavez 96, Chavez 97, PAAM 97]. They have the following features in common.

- **Complexity.** The complexity of the problem refers to the size and the amount of knowledge needed to solve the problem. They are beyond the scope of a single problem solver. Therefore, the problem cannot be solved in isolation.

- **Openness.** The openness of the problem means that the problem is unpredictable and cannot be completely characterised. The purpose and the functionality of the entire system to tackle the problem can not be completely defined when it is designed [Hewitt 85, Hewitt 91, Tokoro 96].

- **Distribution.** This includes the distribution of knowledge, data, information, and activities that are needed to solve the problem. They are inherently spatially, temporally, and functionally distributed. The reasons for this distribution are various that may include the geographic distribution coupled with processing or data transmission bandwidth limitations, the natural functional distribution in the problem, and the desire for distributed control or for modular knowledge acquisition. Other reasons include adaptability, reducing cost, ease of development and management, increasing reliability, and specialisation.

- **Composition.** The compositional feature of the problem refers to the problem-solving system that is usually composed of a number of interacting computational smaller entities. Taking control of cooperation between multiple robots as an example, the current literature survey reveals two main approaches that can be termed as the 'reductionists' approach and the 'constructionists' approach. Reductionism pursues the efficiency obtained by breaking down a big problem into a number of smaller problems to be solved by less powerful entities. Constructionism is seeking to enlarge or to enhance the problem solving ability by integrating a number of less powerful entities into a larger processing unit. In both approaches, the system constructed is composed of a number of computational or acting entities.

Another important and encouraging feature associated with the isomorphic problems is that one problem solving method for a particular problem can have a significant impact

on other problems. To this research it means a particular cooperative framework for multi-robot systems can have a general significance on all the problems in DAI. Additionally, other cooperative frameworks and problem solving methods in both natural and artificial intelligent systems can be adapted to develop the cooperative framework for multi-robot systems.

Returning to the domain of cooperation between multiple robots, it is proper to state here that what is meant by 'a multi-robot system' is different to a conventional cooperative multiple robots' system [Paljug and Yun 93]. In the latter the robots are controlled by a centralised control model. A typical example is a 'car painting' system, where multiple robots, each perform a different task such as agent applying, sanding, painting and polishing. They are controlled by one central computer. This centralised control model is used to resolve a set of dynamic equations, which represent a set of kinematic constraints for each robot in the system [Zheng and Luh 86, Zheng 89, Roach 87, Guptill and Stahura 87, Bejczy 93]. This is an active research area of robotics but is not an area this research is concerned with. In a multi-robot system, the robots are stand-alone performer and they need to be able to reason about their environment, including the beliefs, actions and plans of other robots. They need also to be able to communicate, negotiate and interact with each other to work cooperatively as a community. It should be straightforward for robots to join or leave the community. All of these considerations favour a model in which responsibility is given to individual robots, rather than a central controller. The research lies naturally in DAI. Gasser states:

> "*Researches in DAI are concerned with understanding and modelling action and knowledge in a collaborative of enterprises.*" [Gasser 91]

These collaborative enterprises are called *societies*. A member of such a society is called an *agent* who constitutes the basic processing unit of a system. The agents in DAI are

different from ones in classical distributed systems in a sense that they are *intelligent*. In spite of the debate concerning the nature of intelligence [Minsky 86], many researchers in DAI support introducing attributes of cognitive concepts, such as intentions and beliefs, into agents [Dennett 87, McCarthy 79, Moor 85, Cohen and Levesque 90(1), Rao and Georgeff 91, Shoham 89, Singh 94]. By doing so, it enables researchers to understand, characterise, and analyse the behaviour of agents. The notion of *intentionality* is continuing to emerge as an influential theoretical concept and being used as a basis for a theory in DAI, which is called intentional theory [Cohen and Levesque 90(1), Levesque 90, Jennings 92, Castelfranchi 90, Wooldridge and Jennings 95]. A number of terms which carry everyday definitions e.g. motive, intention, taste, etc. and therefore may be ambiguous are defined with respect to their use in this thesis in a glossary in Appendix E.

However, literature study reveals that the intentional theory, which introduces pseudo-mental terminology into agents in the field of DAI, e.g., an intentional stance using McCarthy and Dennett's term or Newell's knowledge level, does not solve all the distributed computing problems. There are still some remaining problems and some new ones:

- BDI (Beliefs, Desires and Intentions) based agents represent the environment by beliefs and based upon these the agent chooses its action. When an agent represents other agents, it must represent their beliefs, desires and intentions. The other agents have beliefs about this agent, its beliefs and representations of them, ad infinitum[1]. This raises a number of questions such as how to control its convergence? What is the essential content of this recursive representation?

[1]Such a recursive representation shows the nature of cognition. However, it is impossible to be realised. The current approach is based on a first-order representation: another agent is the same as one's representation for oneself. The representation can then be refined via perception and interaction.

- Believing that there are other agents in the environment will cause an agent to act differently[2]. Is there a rational act and what is a rational act anyway? How can the rational act be achieved?

- In a social context, intentional notions, based on a first-order representation and cognitive economy, tend to be inadequate in explaining an agent purposefully, spontaneously and autonomously self-organising. This is because that organising a group is not a productive action[3], it is an intermediate act to affect on the environment. What is needed to motivate agents to organise a team? What role should an agent play in the team? What kind of behaviour should an agent adopt to keep its beliefs consistent and to benefit the team? Overall, an extension of belief revision and nonmonotonic reasoning to a group of agents is needed.

- Extending this anthropomorphic terminology from a single agent to multiple agents raises the question about how the entire system will behave. It may be desirable in the research where a multi-agent system is used as a tool to simulate the social behaviours of the system. However, in industrial applications, like multi-robot systems, simulation is not enough, and a more reliable and controllable scheme is needed.

In summary, the initial motivation of this research is to question the suitability of the intentional theory to DAI systems and to supplement any theoretical supports should they be required in the development of a model of multi-agent cooperation.

---

[2] This can be explained by an agent's belief that the environment changing is due to the actions of other agents. This is the cognitive economy viewpoint, because it is much simpler than trying to cope with a random and unpredictable environment [Huhns, M. in Singh 94 Foreword].

[3] A productive action refers to the idea that the agent's action should be consistent with its beliefs, and has a direct effect on the environment and brings benefit to the agent.

1.3 Research in DAI

There are two main subfields of research in DAI that can be viewed as occupying the two extremes of a spectrum. Distributed problem solving (DPS) [Smith and Davis 81, Findler and Gao 87, Decker 87] lies at one extreme while cooperative problem solving (CPS) [Wooldridge and Jennings 94(2), Demazeau et al. 90, 91, 92] lies at the other.

1.3.1 Distributed Problem Solving

The goal of distributed problem solving is to create a team of cooperating agents that act together to solve a single problem, such as a distributed vehicle monitoring testbed [Lesser and Corkill 83] or monitoring a network of sensors (HEARSAY II) [Erman 80]. Normally this single problem can be decomposed into a number of sub-problems. The sub-problems can be solved by carefully designed and engineered agents or a group of agents. All interactions, such as coordination or cooperation, are incorporated as an integral part in the design of a system on the whole. Therefore, through individual agents' efforts and their interactions with one another the problem can be solved. DPS can be therefore viewed as a top-down designed system.

1.3.2 Analysis of DPS

In theory, DPS represents an answer to the naive question: "How should a task be divided into subtasks so that they can be performed concurrently in an efficient way?" [Tokoro 96]. A system constructed using the DPS approach is regarded as a closed system in this research as the purpose and the functionality of the entire system can be defined when it is designed. The DPS approach in this system is to look for parameters that maximise a value obtained from an evaluation function, therefore it relies on the boundary of the problem and the functions of the system being well defined. The dominant philosophy in DPS in general is reductionism achieved by decomposition. The

reason for this is that the research assumes that agents have only incomplete knowledge and functionality in a problem domain. The theoretical research is focused on organisation, both concurrent and parallel. Therefore, DPS approach has the advantage of being implementable. Actually, it has enabled many useful application systems to be developed [Lesser and Corkill 83, Erman 80]. In summary DPS can be viewed as trying to solve *large* problems[4] in an efficient way.

However, DPS fails to exploit the full potential of the multiple agents' paradigm. It has a few disadvantages if it is put into a wider range of problem domains, e.g., open systems or multiple tasks. These possible disadvantages are listed below:

- System structural arrangement

  In DPS, the overall structure of a system is defined by the designer of the system when the system is constructed. The structure can be centralised, a master-slave model, or hierarchical. The purpose of the structural arrangement is to ensure the avoidance of any potential conflicts, deadlocks and resource starvation at the system level. This systemic structural arrangement is static and inflexible. Once the system is put into application, it is difficult to modify.

- Limitation of agent's autonomy

  The role of an individual agent in a system is pre-defined by the system designer. The specific assigned roles and the higher level systemic restrictions add extra constraints to individual agents. It appears that the agents cooperate to accomplish a given task in the overall system, but actually the agents themselves are only role players and are not aware of other roles. Therefore, an individual agent cannot be said to be a real autonomous agent in the sense of general

[4] Here *large* means the problem is too large or too complex to be solved by a single agent.

autonomy[5]. On the other hand, because of the dominance of the holistic view, the majority of these systems adopted the benevolent assumption[6]. In such cases, the agents' predisposition is to be helpful to others, even to the detriment of their own problem solving in some cases. This is also erosion of their autonomy. Hence the agent in a systemic structural arrangement is not a social agent, and a number of agents situated in this structure are not a social team either. This is because the individual agent does not have a consciousness of the existence of others and therefore cannot interact with them intentionally.

- The agenthood is weak

DPS does not have a notion of a *group* instead the agent in DPS has a notion of *self*. This means that the overall goal of the system is not known by all the members and the individual agents pursue different goals for their own benefit, instead of the common goal of the entire system. If an agent adopts a benevolent stance, then it may detract from its own problem solving process to respond to another agent's request for help and thus detract from the system as a whole.

- Failure to deal with multiple tasks

DPS is insufficient to deal with multiple goals or multiple problem-solving plans. In DPS, there is no united worthwhile notion of an overall system making a decision about which goal to adopt and which plan to execute and who is going to do what and when.

- It is not sufficient for an open problem

[5] General autonomy here means that an agent should and can decide for itself which goals to adopt, how the goal should be pursued and which action should be performed and when.

[6] That is an agent is willing to perform all tasks requested by its acquaintances and volunteered its services to others.

Based on a premise of a closed problem, systems built with DPS approach have a clear boundary and functions of the systems. They restrict the system's ability to handle changes in the architecture of the system. Another disadvantage of DPS is that if the central computer or master computer fails to perform, if the structure is centralised or in master-slave model, then the whole system fails.

1.3.3 Cooperative Problem Solving

In contrast to the systems with DPS approach, systems with CPS approach are not restricted to solving a single problem or to performing a single task. The most significant difference lies in the individual agents. In CPS, agents are autonomous and mostly pre-existing. The research is focused on how to achieve a co-ordinated intelligent behaviour, and possibly cooperation or competition among a collection of autonomous intelligent agents. Although an agent in CPS can be a specific task performer or a particular expert, situated in an agent's society, it should coordinate its knowledge, belief, skills, and plans not only to achieve its own goals, but also the goals of the society as a whole [Bond and Gasser 88]. Since agents in CPS are designed first, CPS can be viewed as a bottom-up designed system.

1.3.4 Analysis of CPS

CPS can be viewed as trying to answer the question, "Can each agent obtain higher benefits by working as a group rather than working in isolation?" [Tokoro 96]. This type of agent differs from the agent in DPS in the respect of having a notion of *society*. The dominant philosophy in CPS is that the success of the individual is the sole metric for the performance evaluation of the system. The research is very much focused on the following: what is an individual agent? What kind of attributes should be included in an agent? How should an agent act in a multi-agent environment?

The practical advantages of this approach are that it narrows down the problem scope, and reduces the amount of design at the level of the overall system. It has the theoretical advantages of being able to include many other developed theories such as, Game Theory, Biological Evolution, Ethnology, Economic Sociology, and so forth. However, this agent-centric view also has disadvantages:

- Problem solving strategy

  Since the individual agents have been intensively focused, they are often designed to maximise their individual benefit or to maximise their individual utility and adopt a self-interested stance. When placing this kind of agent in a social context to form an agent's community, any kind of possible social aspects may be destroyed.

- The notion of organisation is weak

  Because the spirit of CPS is to cope with dynamic situations and openness, its bottom-up design strategy usually emphasises the bottom and ignores the top. In the real world an agent cannot exist in isolation. Agents are inevitably interdependent, interacting, and influencing with each other to some degree. Lack of an organisation notion [Ndumu 97] will lead self-interested agents into anarchy. They will struggle for bounded resources such as time, space and power even if this behaviour detracts other's benefit or the community's performance.

- Non-predictable system behaviour

  The overall multi-agent systemic behaviours in an agent-centric approach emerge from the interplay between the agents. Therefore, it is extremely difficult to predict or control a larger system's behaviour or performance. This might be desirable for social simulation and multiple computer system's simulation

> [Gilbert 94]. But it is obviously not desirable for industrial and commercial application. Nobody can afford a "black art" [Jennings and Campos 97, page 12] kind of solution to a real problem.

Many researchers have noticed the downsides of the agent-centric approach [Jennings 97, Wooldridge 96, Tokoro 96]. A significant proportion of the current research in the field of CPS is focused on trying to attain socially responsible behaviour in an agent. This has been identified by Hewitt and Gasser [Hewitt 91, Gasser 91] as a key problem in the area. To date, a wide variety of solutions have been provided. They include the redefinition of the attributes of an agent, e.g., Jennings' social level [Jennings and Campos 97], meta-level information exchange, communication, behavioural law, decision making, negotiation, and so on. All these researches are still starting from and based upon individual agents, taking the social context into account to pursue an agent's sociable behaviour. Unfortunately, they only emphasise the agent's cooperation model but not the organisation model.

To sum up, there is reductionists' work formed in DPS, which is insufficient to handle open problems. Its system level arrangement detracts from an agent's autonomy. Although the holistic view in DPS may influence the design that enables benevolent agents to be adopted, it fails to exploit the full potential of a multi-agent system. However, its fixed structural arrangement has the advantage of implementation, evaluation and analysis of system performance. On the other hand, the constructionists' based work, in the CPS approach, aiming for open systems emphasises individual autonomy. It does not have an agent's systemic concept because the system is open. The consequence of this is that its overall system performance is unpredictable. Eventually, its implementation becomes something of "black art."

1.3.5 What is Missing in Existing Approaches?

Up until now, many researchers have noted that it is interesting to view agents as *artificial lives* [Langton 93, 94]. In comparison with humans and human society, two things are missing in the current DAI research.

1. **An explicit model to represent the social characteristics of an individual agent.** [Moulin and Chaib-Draa 96]

   Miller and Rice said:

   > *"An individual has ... no meaning except in relation to others with whom he interacts. He uses them, and they him, to express views, take action, and play roles. The individual is a creature of the group, the group of the individual."*
   > (Quoted from [Miller and Rice 67] page 17)

   This may be a one-sided and arguable statement, but it does reveal the importance of social characteristics of an individual agent in a social context. A human, as a creature of the group, inherently possesses the characteristics of perception and involvement towards one or more groups by bearing groups' norms and values. Its corresponding concept in theory is described in relationship with other individuals. This relationship is further described by a notion of organisation [Miller and Rice 67], such as mother and me, the family, class, school, and so forth. As described in the previous subsection, the current approaches have insufficient representation of this social organisational characteristic in an individual agent's model. In DPS approaches, the agents under the systemic arrangement are only role players. They do not have a built-in model of organisation. The role of an agent is pre-defined by the system designer to facilitate a particular design purpose. The agent itself has no concepts of organisation and the existence of other agents, and thus can not *interact* and *use*

them intentionally. A group of agents may work cooperatively nevertheless the cooperation is again a systemic arrangement. In CPS approaches, agents have been intensively focused. They are more likely to adopt a self-interested stance, although this self-interested stance does also not exclude an agent's social characteristics. However, its social characteristic is represented integrally in the agent's conventional action model, which is a first-order intentional model [Dennett 87, McCarthy 79, Cohen and Levesque 90(3)] that models an agent's action with *beliefs* and *goals*. Under the dominant philosophy of maximising an agent's individual utility, the first-order intentional theory is insufficient to explain the agent purposefully self organising or involving an organisation. Cohen and Levesque noticed that an action in a social perspective is grounded in the actions of many agents' activities *taken together*, it is no longer a matter of individual choices [Cohen and Levesque 90(3)]. Gasser also indicates that the current CPS approaches have to change to adopt the social perspective [Gasser 91]. He explains the reason of this change is that, "Many concepts associated with individual agents are, in sociological terms, reifications constructed through joint courses of actions and made stable by webs of commitment, or alliances among the agents using them." Therefore a separate and explicit model of organisation is necessary for an agent's social behaviours.

2. **The concept of organisation itself as a dynamic intermediate granularity between *an agent's group* and *a group of agents*.**

   As described in the above subsections, *an agent's group* refers to the systemic arrangement in the DPS approaches where agents are incorporated as an integral part in the design of the whole system. If the system on the whole can be viewed as an agent's organisation, this organisation is designed by the system designer

when the system is constructed. It is a top-down design approach, and thus it is static and inflexible to a large degree. Once the system is constructed, it is difficult to change. It does not support automatic scaling-up of the systemic functionality. *A group of agents* refers to a collection of agents in the CPS approaches. Although its bottom-up design strategy is useful for solving the systemic scale-up problem, unfortunately the scale-up is based on an emergence from the interplay between agents. It has the problem of convergence since there is not a control scheme to ensure the systemic convergence and to prevent anarchy. An intermediate organisation notion is necessary, lying between the static systemic arrangement and the non-controlled free merging. It can be viewed as a controlled bottom-up scheme where the function of its components, rather than an individual itself is defined. The systemic convergence can be controlled by the manner of its components' merging. Therefore it enables study of the features of the organisation, such as its formation, functionality, communication vocabulary, languages, protocol, and problem solving coordination method. Its overall systemic performance can be controlled and evaluated. In this way, the bottom-up design approach with CPS technology will no longer be a form of "black art."

1.4 Original Contributions

This research is concerned with developing a framework for multi-agent cooperation particularly for cooperation between multiple autonomous, pre-existing robots. Four complementary efforts are represented in the thesis:

- Construction and implementation of a test-bed

A parameterised multi-agent cooperation test-bed is built. It is used to implement the existing multi-agent cooperation frameworks to identify their shortcomings, which in turn set the objectives for the new cooperation framework. The test-bed is also used to test the theory and the model developed in this thesis against their claims. Finally the implementation of the framework for control of multiple robots' cooperation is built, based on this test-bed. It also provides guidelines for other applications.

- Development of a novel cooperation framework

With an understanding of the features of the problem and the necessity of a new approach to the problem revealed by empirical experiments in implementing the existing cooperation frameworks, a multi-agent cooperation framework called Shifting Matrix Management (SMM) is then proposed. It is a refinement and an integration of the DPS and the CPS approaches inspired by organisation theory and economic team theory. Within this framework, a dynamic, temporary organisation of multiple agents in the form of a team is constructed. Tasks for the system are accomplished in the form of teamwork.

- Development of a quantitative action theory

A quantitative action theory is developed as a supplement to the existing intentional theory. It is identified in this research because the intention theory does not address the choice of an agent to realise their intention. The quantitative action theory enables a more complete, comprehensive and rational model of multi-agent cooperation to be developed, because the theory explains why an agent chooses and adopts a particular way to achieve its intended goal and to what extent an agent ought to insist on its choice of action.

- Development of the SMM model

  The SMM model is a formulation of the SMM framework. It is developed to make the theory mathematically tractable and to map the theory to its implementation. It also provides guidelines for designing any similar multi-agent cooperational implementations.

1.5 Thesis Outline

Chapter 2 presents the construction of a test-bed. It is based on in-house software called Algorithmic and Rule-based Blackboard System, ARBS for short. The three purposes of building the test-bed can be identified as: 1) gaining the first experience of building a multi-agents system and thus developing an understanding of the practical issues around an application system, 2) enabling different cooperative frameworks to be implemented, studied and evaluated, 3) using it to synthesise existing approaches for identifying theoretic requirements to support the development of a general multi-agent cooperation model.

Chapter 3 and 4 report practical experiments of existing cooperative frameworks. It is the first method adopted in this thesis to develop a new multi-agent cooperative framework, which is classified as *learning by doing*. Two influential cooperative frameworks named the Contract Net and the CPS framework are introduced, implemented and tested respectively on the test-bed to study their applicability in the problem domain. The implementation and tests are aimed at identifying shortcomings of the frameworks and objectives for the new cooperative framework. Chapter 3 reports the implementation and tests of the Contract Net framework, and Chapter 4 reports the implementation and tests of the CPS framework.

Chapters 5, 6 and 7 represent the main theoretical contribution of this research. The SMM framework is proposed in Chapter 5. It is proposed because of problems perceived in the implementation of existing cooperative frameworks and draws upon ideas from other fields. The latter is the second method adopted in this thesis which is classified as *learning by analogy*. The SMM framework inspired by organisation theory identifies six stages in a cooperation process. This six-stage framework sets up general objectives for a theory development.

Chapter 6, as the second part of theoretic contribution, provides a quantitative decision theory. It is developed because the SMM framework identifies that both *intention* from intention theory and *social dependence* from behaviour theory are inadequate in explaining an agent's choice between alternative actions in achieving an intended goal. The quantitative decision theory reveals an agent's decision on actions depends on another mental state named *preference*. It is represented by three quantitative functions that the agent adopted.

Chapter 7, as the conclusion part of the theoretical contribution, develops a general and abstract model. This can cover a wider range of applications by serving as top-level and abstract specifications for building multi-agent systems.

Chapter 8 provides empirical evaluations of the SMM model. Taking control of cooperation between multiple robots as an example implementation of the SMM model. A number of controlled experiments are under taken, varying the impact of the variables in the SMM model on the systemic behaviour. Comparison between the SMM model and the other models, which were initially investigated, are taken to test the benefits of the SMM model. The tests' results are reported as proof of the claims.

Chapter 9 provides a summary of this research with some concluding remarks. Several ideas for further investigation are given.

Chapter 2

Developing a Multi-Agent System

"Starting from scratch is only trying to emphasise the difficulty of a job. It is not always true in a sense that the job is part of other larger job which has been put forward by someone else; the tools used in doing the job have been invented by someone else; the way of doing the job has been proved by success or failure of many other similar jobs; ... After all, any task is part of a larger task if you come out of your working domain. So, be wise, never say start from scratch, say how you are using the results from others' work."

-- A Chinese proverb

This Chapter describes the development of a multi-agent system. It has the following three objectives:

1. Helping to develop an applicable and simple-to-use framework for control of multiple robots' cooperation.
2. Functioning as a test-bed that enables different cooperation schemes to be implemented, studied and evaluated.

3. Acting as a means to synthesise existing approaches, to identify theoretical requirements to support the development of a general multi-agent cooperation model.

The chapter is arranged in the following manner: section 2.1 describes an in-house software system adopted in building the multi-agent system. Section 2.2 describes the representation of agents in the system. Section 2.3, introduces a new paradigm which is implemented in this multi-agent system and enables different cooperation schemes to be studied. Finally, section 2.4 provides the construction of the test-bed with control of multiple robots' cooperation as an implementation of the multi-agent system.

2.1 An In-house Software - ARBS

ARBS - Algorithmic and Rule-based Blackboard System, is a blackboard system developed at the Open University initially for building different knowledge-based systems [Hopgood 93, 94, 97]. It is a general toolkit since it enables many other knowledge modules such as rule-based, procedural and neural networks, and also enables many different inference mechanisms such as forward chaining, backward chaining and "hypothesise-and-test" [Hopgood et al. 93, Hopgood 97]. The variety of options and the flexibility of ARBS are the main reasons why it is adopted in developing the multi-agent system.

2.1.1 Overview of ARBS

ARBS is based on the blackboard model [Hayes-Roth 85, Engelmore and Morgan 88]. In this, a problem, represented by a certain model, and its solution or part solution, evolves in a global memory area, which is called the blackboard. Knowledge about the problem is divided into modules, which are called knowledge sources (KSs). Each rule-based KS contains a knowledge base and an inference engine, which enables the

knowledge to be applied towards a final solution for the problem from the current state. Problem solving is an incremental process, where each KS reads information from the blackboard, adds new information to the blackboard and deletes old information as shown in Figure 2.1.

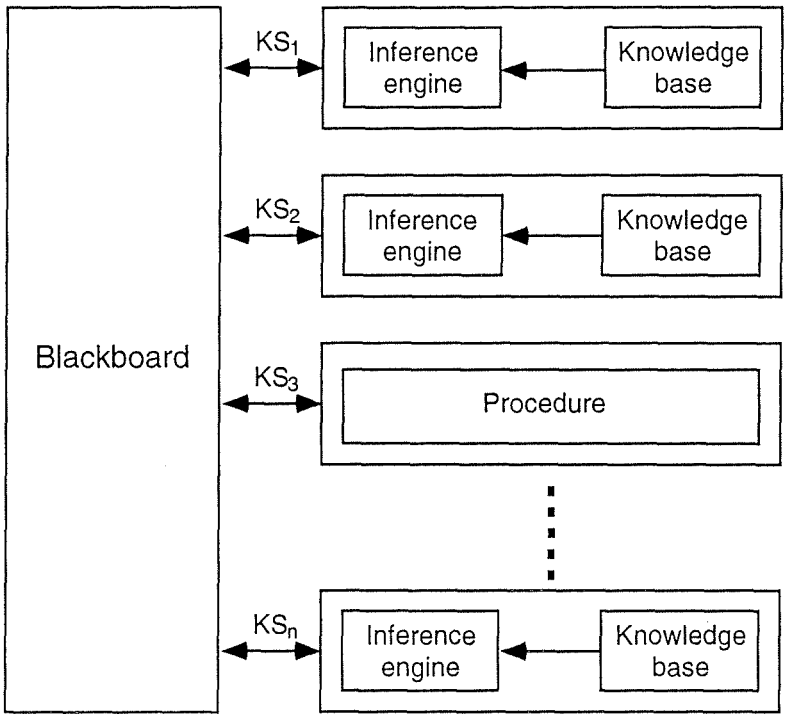


Figure 2.1 ARBS Architecture [Hopgood 93]

2.1.2 Knowledge Sources in ARBS

Each KS in ARBS is contained in a record. Its structure is shown in Figure 2.2. There is a set of preconditions, in the *preconditions* field, which specifies the conditions that must be satisfied before the KS can be activated. There is also an *activation flag* field which can be set to switch the KS on or off. ARBS has a control model which is similar to forward chaining within a rule-based system. It examines each KS in turn, testing the preconditions and activating the KS if the preconditions are satisfied. This is the simplest strategy based on "first come first served" forward chaining [Hopgood 93]. More sophisticated strategies can be applied to the selection of knowledge sources if it

is necessary. There are three basic types of knowledge source in ARBS. They are procedural, rule-based and neural networks.

When a KS is activated, it applies its knowledge to the current state of the blackboard, adding to it or modifying it. The entry in the *KS type* field states whether the KS is procedural, rule-based or a neural networks KS. If a KS is rule-based then it is essentially a rule-based system in its own right. The *rules* field contains the names of the rules to be used and the *inference mode* field contains the name of the inference engine. When the rules are exhausted, the KS is deactivated and any actions in the *action* field of the KS are performed. These actions usually involve reports to the user or the addition of control information to the blackboard. If the KS is procedural or neural networks', then the required code is simply included in the *action* field and the *inference mode* and *rules* fields are not used.

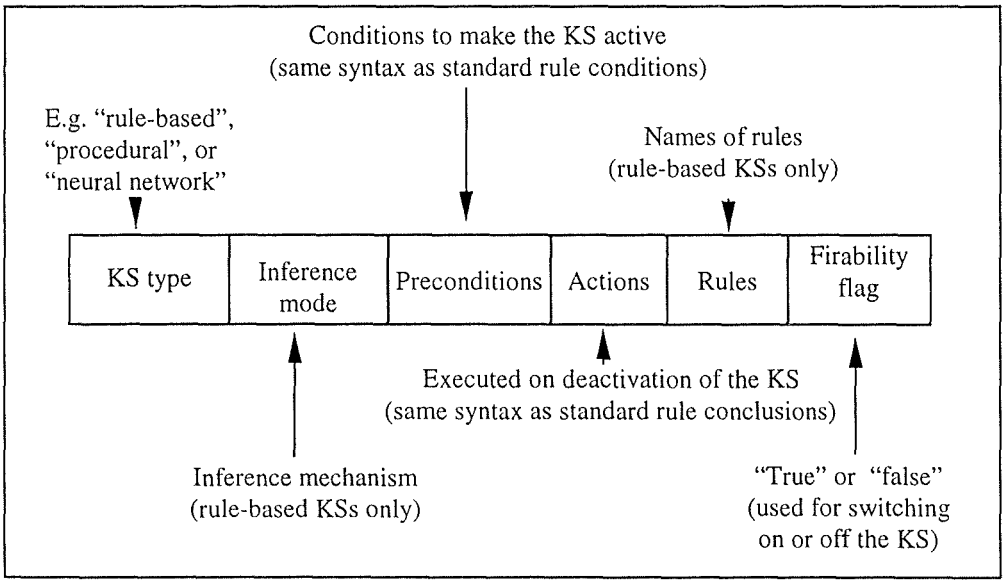


Figure 2.2 A knowledge source in ARBS [Hopgood 93]

The blackboard architecture is able to bring together the most appropriate KS for handling specific tasks [Nii 86]. Procedural tasks are not entirely confined to procedural KSs, since rules within a rule-based knowledge source can access procedural code from either their condition or conclusion parts, which can be seen from the next section.

A knowledge source in ARBS has flexibility with regards to its size and complexity as well as type. A KS in ARBS can vary from simple structures that have a single procedure to large rule sets, which access many procedures and functions.

2.1.3 Rules in ARBS

Rules are used in ARBS in two contexts:

- to express domain knowledge within a rule-based knowledge source; and
- to express the applicability of a knowledge source.

Just as knowledge sources are activated in response to information on the blackboard, so are the individual rules within a rule-based knowledge source. The main functions of ARBS' rules are to look up information on the blackboard, to draw inferences from that information, and to post new, or delete old information on the blackboard. Apart from procedural KSs, procedures and functions can also be directly accessed from within the condition or conclusion parts of any rules. This would typically be for numerical calculations, external communications or database lookup.

Each rule is a list comprising four elements in ARBS: a number to identify the rule, a condition part, a rule operator "implies" and a conclusion part. A rule is therefore defined as follows:

Rule:: [number *condition* **implies** *conclusion*],

where the condition may comprise subconditions joined with Boolean operators AND and OR. The depth of nesting of Boolean combinations of condition statements is limited only by their understandability. The conclusion can contain any number of statements. Atomic conditions that do not contain any Boolean operators can be evaluated in any of the following ways:

- test for the presence or absence of, and look up, information on the blackboard;
- run algorithms which may return numerical or Boolean results;
- numerical comparison of variables, constants, or algorithm results.

The conclusions, or subconclusions, can comprise any of the following:

- add to or remove information on the blackboard;
- call procedures, optionally adding the results to the blackboard;
- report actions to the operator.

Whilst this is a fairly simple philosophy, it provides a flexible and powerful rule structure in which complex conditions and conclusions can be constructed. The flow of information between different parts of the same rule is made by an extensive use of local variables. In contrast, rule-to-rule communication is carried out through the blackboard.

2.1.4 Blackboard in ARBS

In ARBS, the blackboard is made up from a *list* in the Pop-11 language. Therefore, adding or deleting a partition of blackboard is implemented by adding or deleting a sublist from a list which is the blackboard itself. Posting or removing a statement from a partition of the blackboard is also implemented by adding or deleting an elemental list from a list. The retrieval of information from the blackboard is realised by the pattern-matching facilities provided by Pop-11. It can be demonstrated by considering a rule:

```
[101    ;;; the rule number
        [
                [present  [task is ?task] task_generation]
        ]
        implies
        [
                [report [task appears on the blackboard] nil]
                [add [task is ~task] task_to_bid]
        ]
].
```

This rule examines the partition of the blackboard called task_generation. Like all the blackboard partitions, task_generation is a list. Supposing that task_generation contains the sublist:

[task is [move [discs a b c] from piles1 to piles3]],

then the condition of the rule is true, and the local variable task would become bound to list [move [discs a b c] from piles1 to piles3]. This simple condition has thus checked for the presence of information on the blackboard and retrieved information. The first conclusion of the rule reports the deduction that *task appears on the blackboard* to the user to indicate that the system is now processing the task. The second conclusion of the rule involves adding information to a new partition named task_to_bid of the blackboard. Notice the use of the '~' symbol, which instructs ARBS to replace the word task following the symbol with its current value which is a list.

2.1.5 Inference Engines in ARBS

The strategy for applying rules is a key decision in the design of a system. ARBS provides a greater flexibility by allowing different inference engines. The inference engines provided by ARBS include forward chaining, backward chaining and "hypothesise-and-test." The first two inference engines for rule-based KSs involve using either single or multiple instantiation of variables in ARBS. ARBS also has provision of using the hybrid inference mechanism which is an inference mechanism that can be thought of as part forward chaining and part backward chaining [Hopgood 93]. The provision can construct a network, prior to running the system, representing the dependencies among the rules to support the hybrid inference mechanism. These different inference engines provide a flexibility that enables knowledge sources to use whichever inference mechanism is most appropriate.

To sum up, ARBS is based on the blackboard model with variety of knowledge modules and flexible inference engines. A blackboard system is analogous to a team of experts who communicate their ideas through a physical blackboard, by adding or deleting items in response to the information that they find there. Each knowledge source represents such an expert having a specialised area of knowledge. Knowledge sources are applied in response to information in the blackboard, when they have some contribution to make. This leads to increased efficiency since the detailed knowledge within a knowledge source is only applied when that knowledge source becomes relevant. Besides the modularity and the flexibility, ARBS also inherits features from the blackboard model such as multiple participants, opportunism in knowledge application and incremental problem solving. All the above features make it an ideal in-house software to build a multi-agent system.

2.2 Representing Agents in ARBS

An agent as a basic entity of a multi-agent system has to be defined and constructed before the multi-agent system can be constructed. Because ARBS provides great flexibility in both the modularity and grain-size of KSs, the model and the structure of an agent which is to be represented in ARBS are not necessarily fully developed. Any change in an agent's model can easily be represented by a simple modification to the KSs. This is an important feature for the system and the test-bed that is to be built.

2.2.1 An Agent's Model

Modelling an agent and designing an agent architecture based on the model is an important research topic in the agent and multi-agent field [Wooldridge and Jennings 95]. Maes defines an agent architecture as:

"[a] particular methodology for building [agents]. It specifies how ... the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions ... and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology." (Maes 91, p115, quoted from [Wooldridge and Jennings 95])

The agent model adopted in this thesis relies heavily on the work presented in [Li et al. 97]. The architecture of an agent is shown in Figure 2.3, which is a hybrid architecture that integrates both deliberative and reactive architectures. A deliberative architecture is also called a belief-desire-intention (BDI) architecture. It contains a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI. Some examples of this architecture include *STRIPS* [Fikes and Nilson 71], *IRMA* [Bratman et al., 88] and *GRATE** [Jennings 93b]. In contrast, a reactive architecture, which is capable of reacting to events that occur in the environment without engaging in complex reasoning. Examples include *Subsumption Architecture* [Brooks 86], *PENGI* [Agre and Chapman 87], *Situated Automata* [Rosenschein 85] and *Agent Network Architecture* [Maes 89, 91].

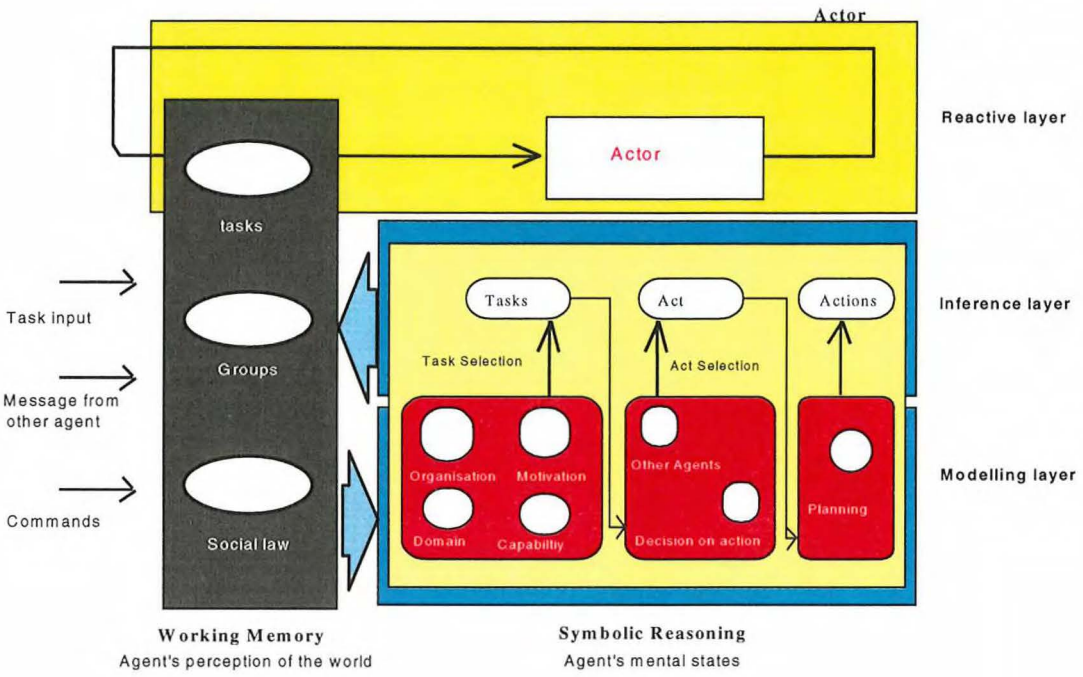


Figure 2.3 A hybrid agent architecture

The architecture shown in Figure 2.3 comprises three components: working memory; symbolic reasoning; and an actor. A working memory represents the agent's perception of the world. In a multiple agents' world any one agent's perception of the world is a subset of general world, so the same world can be interpreted differently by different agents. An agent's perception of the world includes social and cooperation laws, the existence of other agents and the current state of agents who exist in the agent's world. The symbolic reasoning represents the mental states of an agent, which is a two-layered, multiple processes reasoning device. Within this symbolic reasoning component, an organisation model is explicitly represented. The actor component represents the executable functionality of an agent. It performs the function of the agent's reactive behaviour, which is reacting to events that occur in the world without engaging in complex reasoning.

2.2.2 Representing Agents in ARBS

An agent's architecture as shown in Figure 2.3 demonstrates what is needed for an agent in order for it to behave appropriately in the performance of a task, which may involve working cooperatively or working in isolation. The behaviours that are performed by an agent when it is involved in a task's performance, include the following as a minimum:

- perception,
- goal selection,
- act selection,
- planning, and
- action.

These behaviours reflect the separated steps[1] of a task's performance. If an agent is involved in the entire performance process of a task, then it needs to perform all of the behaviours identified in each step. However, a task's performance process may not always be the same or an entire process as stated above. Some tasks' performance may have fewer steps. The involvement in a task's performance by different agents may also be different because of agents' ability and the behavioural strategies adopted by the agents. One agent may be involved in a whole process while another agent may just be involved in a few steps. After all, when representing an agent in the context of performing tasks, it is not only the agent's architecture and the strategy adopted by the agent that need to be considered, but also the process of the tasks' performance.

The diversity in both behaviour and strategy applied by individual agents at each step encourages the following representation in ARBS:

[1] Here the term "steps" refers to subprocesses, which can be identified in a task's performance process.

1. The steps in a task's performance are represented by KSs.

2. The agents' behavioural strategies in a step are represented by rules in the KS.

Therefore an agent in ARBS is represented by a set of rules that are spread over the different knowledge sources that represent different steps of a task's performance. The rules and the knowledge sources communicate with other rules and knowledge sources indirectly through the blackboard. To reduce the search time to find a piece of useful information on the blackboard, the blackboard is partitioned into two blocks. They are a public area and the agents' private area. The public area is used to exchange information between agents. The private area is used to represent an agent's perception of the outside world. The representation of an agent is illustrated in Figure 2.4 where agents are represented by horizontal surfaces.

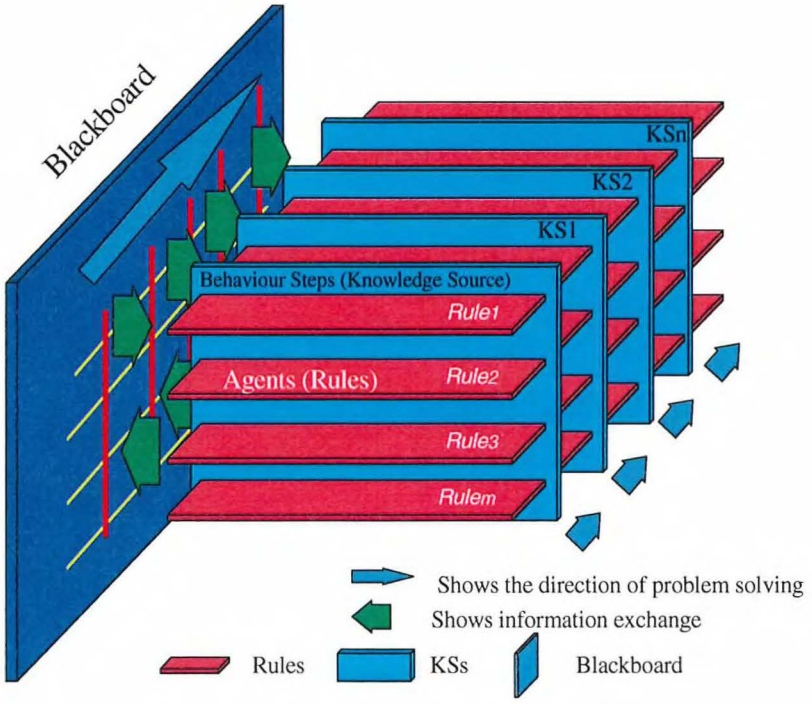


Figure 2.4 Representation of agent in ARBS

This agent's representation has the following advantages:

1. **It separates agents from a task's performance process.** In the above representation, a task's performance is represented by KSs and agents are represented by the rules in a KS, which refer to the agent involved in this step of the task's performance. In ARBS, rules in KSs are separate from the KSs in a sense that adding or deleting a KS will not affect the rules in other KSs. Similarly adding or deleting rules in one KS will not affect other KSs. Different tasks may request different processes during their performance. For example, a *complex task*[2] may require planning, but an *atomic task* will simply need action. Therefore the separation of agents from the process of a task's performance enables addition or deletion of one or more steps in a task's performance without modifying the agents' behaviour in other steps of the task's performance. At the same time changing an agent's involvement in a step of a task's performance by adding or deleting rules in the corresponding KS will not affect other steps in the task's performance. This provides a great flexibility for testing different tasks' performance process and different cooperative schemes, which may involve different agents in different steps of a task's performance.

2. **It improves information exchange efficiency.** Information among agents is exchanged indirectly through the blackboard in ARBS. A task's performance is decomposed into different steps, which are represented by KSs. This enables each step to focus on a particular issue. When a KS is activated, participating agents are particularly focused on this issue through perceptions, actions and contributions to the task's performance. Thus there is less information to be exchanged at each step of task's performance compared with the information to

[2] A complex task is a task type in contrast with other task types such as atomic task and combination task. Their definitions are given in the next chapter.

be exchanged for a whole task's performance. It is also more efficient since the information exchanged is particularly focused.

3. **It enables cooperation at each step of a task's performance.** Since agents are represented by rules in different KSs and each KS represents an individual step in a task's performance, an agent's contribution to the overall task performance is also divided into contributions at each step. This provides the possibility that agents cooperatively make contributions to the specific issue on which the step is focused. For example, instead of cooperation in a procedural manner such as one agent does planning and the other does the action, cooperation can occur in both planning and action.

4. **Easy to change an agent's behavioural strategy.** In ARBS, a rule in a rule-based KS can also access procedural code (see section 2.1.3). Procedural codes are normally written in C or other high-level languages, which are external to ARBS. It is therefore easy to write a different behavioural strategy using a high-level language as external code and allowing different rules to call different external code to realise a different behaviour strategy (see Appendix D).

2.3 A New Paradigm of Multi-Agent Systems

With the above description of in-house software and the representation of an agent in ARBS, it is now time to describe the construction of the multi-agent system that is built to realise the three objectives listed at the beginning of this chapter. The above agent's representation in ARBS enables easy alteration to the number of agents in the system or the number of steps in a task's performance. A new paradigm is introduced to control the change of a task's performance process and the sequence of steps in a task's performance process.

In ARBS, three control mechanisms are provided to activate a KS. They are *firability flag*, *inference mode* and *preconditions* of a KS. The new paradigm makes use of these three control mechanisms together to build a general tasks' performance process by a number of KSs and a particular order of KSs. Since the *inference mode* provided by ARBS is *first come, first served*, this requests that all the KSs and the order of these KSs in a task's performance process has to be settled before the system can be used. In our multi-agent system one purpose of constructing such a system is to test existing cooperative schemes where KSs and the order of the KSs needs to be changed. These dynamics can be realised by using two other control mechanisms provided by ARBS. The new paradigm consists of the following points:

1. A new public panel in the blackboard is set up particularly for KS sequencing.
2. The preconditions of KSs are set-up to ensure that the designed sequential order is executed. Changing the sequential order of KSs can only be made by changing the preconditions of KSs.
3. Adding or deleting agents from the system by adding or deleting appropriate rules in certain KSs.
4. Adding or deleting new steps into a task's performance process by adding or deleting KSs.
5. Switching the firability flag off and on can rest or activate existing KSs to be tested.

With the above new paradigm, a basic multi-agent system is built with the following KSs and activating order. It is illustrated in Figure 2.5. Note that this multi-agent system is a basic system. It is based on the assumption and promises of the new paradigm that adding and deleting agents and steps in a task's performance process are both

convenient and simple. Actually, in existing multi-agent cooperation schemes, they do not have the same task's performance process nor do they have the same order. These differences allow different cooperations to be identified. The multi-agent system shown in Figure 2.5 can easily be modified to implement different cooperative schemes therefore it enables study and comparison to be carried out.

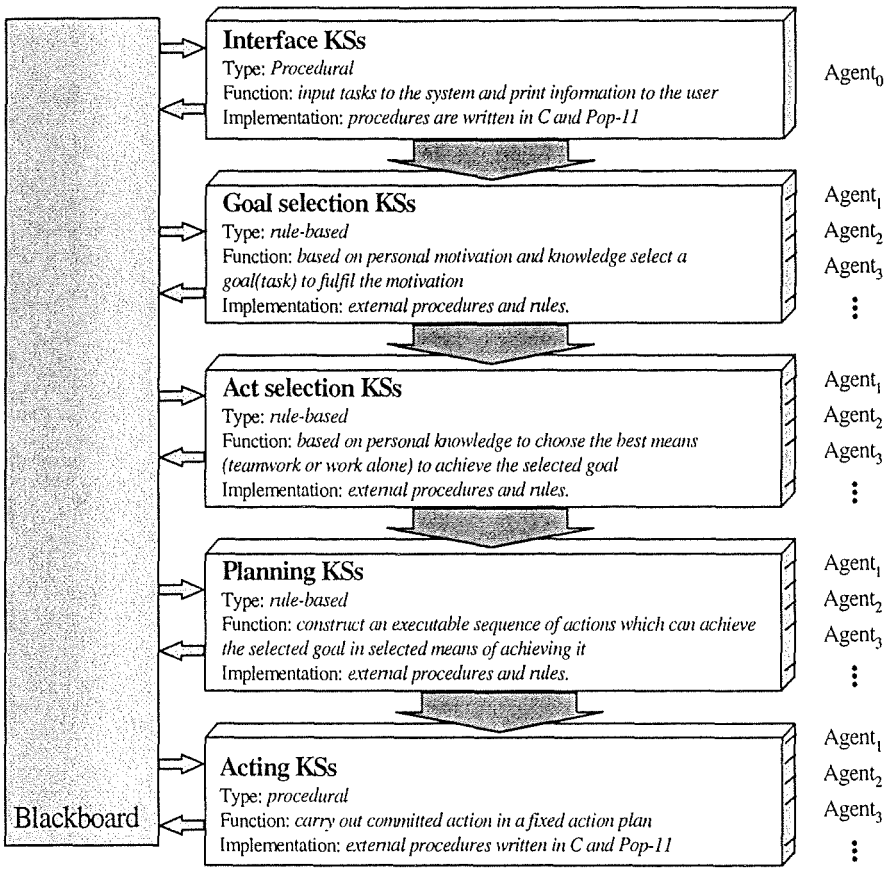


Figure 2.5 A multi-agent system in ARBS

2.4 Control of Multiple Robots Cooperation

This section briefly introduces the implementation of the multi-agent system, which is built in ARBS to control multiple robots' cooperation. This implementation has two purposes. One is to develop a control scheme that can be used to control multiple robots' cooperation. The other purpose is to construct a test-bed that can be used to evaluate existing cooperative schemes in multi-agent systems.

2.4.1 The Robots

The robots available are called MA2000. They are simple robots used for educational purposes. They do not have an in-built sensor system nor mobility[3]. Figure 2.6 is an illustration of one of these robots.

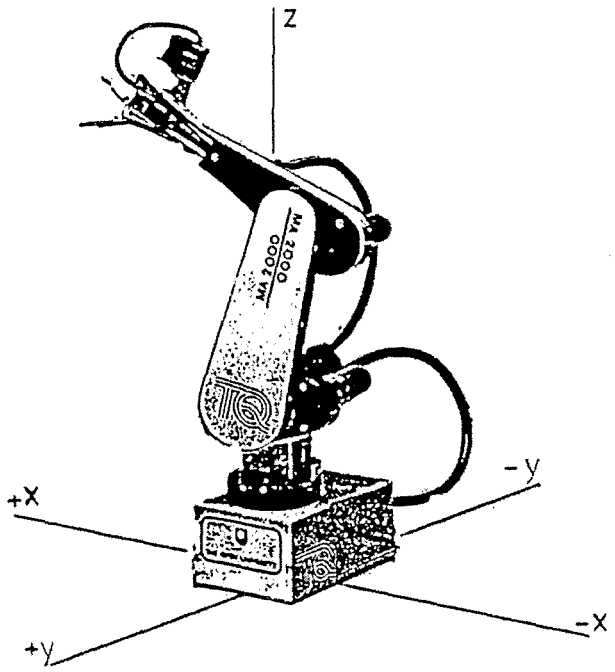


Figure 2.6 MA2000 Robot

Each robot has a control box and a host personal computer. They are used as user interfaces and a central controller for the robot. First, they receive control commands and programs from users or other computers. These commands and programs are then converted into motor driving signals, and finally drive different joints of the robot to move. The robot can perform pick up, move, and put down actions. Each MA2000 uses two coordinate systems to control its movements. The *posture* coordinates system has *Waist, Shoulder, Elbow, Pitch, Yaw, Roll* and *Grip*; the *XYZ* coordinates system has *X, Y, Z, Azimuth, Meridian, Roll* and *Grip*. Four control models are available to drive the robot: *Drive* (by control pad), *Lead-by-nose*, *Continuing path* and *Off-line*. The first three control models are constrained by the kinematic constraints deployed by each

[3] The mobility used here refers to change of the standing position.

physical joint when the robot is in use. The last control model breaks the link between valid coordinates and the kinematic constraints and thus the physical constraints have to be considered when the coordinates are generated. The robot's control models also provide facilities to specify moving speed and conditions.

In the test-bed developed in this research, two robots are used[4]. They are distributed and autonomous in the sense that data processing and signal control are generated by separate PCs. The *off-line* control model is used because the coordinates of movement are generated by a third party computer and not the host computers. Thus the kinematic constraints of each robot are stored and considered as part of the robot's inherent constraints when representing an agent's deliberative behaviours. For example, when an agent representing a robot is making a decision on certain movements, it ought to be concerned with whether the movements are possible or not. Both *posture* and *XYZ* coordinate systems are used in the system.

2.4.2 The System Layout

The multi-agent system in control of multiple robots' cooperation consists of a SUN SPARCstation where the ARBS based multi-agent system is running, a communication channel, two PCs and two MA2000 robots. It is shown in Figure 2.7.

[4] More robots can be used subject to availability. However this research defines that the relationship between robot and agent is one to many. This means that one robot can be represented by many agents. In this case the robot is the manipulator for many agents.

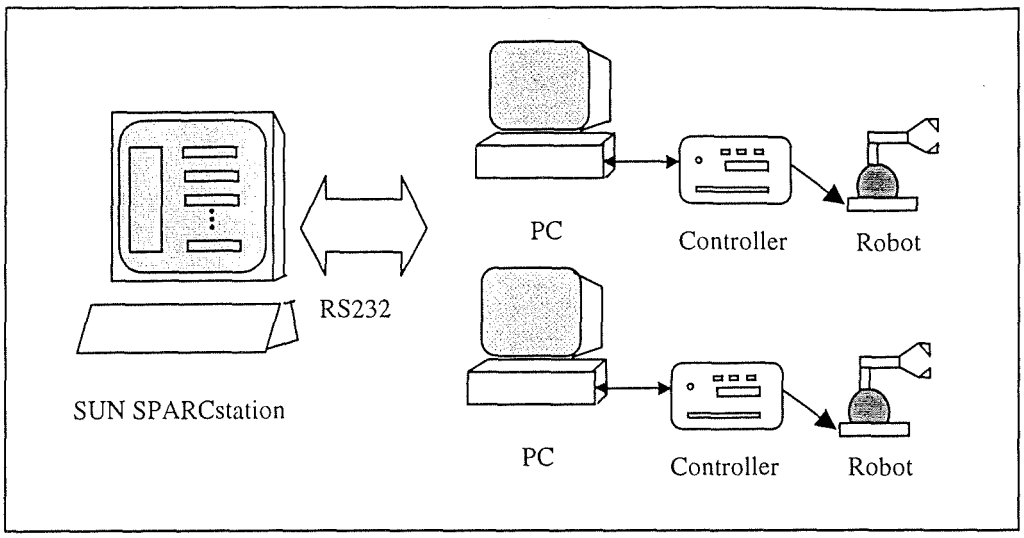


Figure 2.7 The system layout of the multiple robots' cooperation

From Figure 2.7, the whole cooperative system is built on top of the existing robot's system. There is no modification to the existing robot's control system. The communication between the SUN and the PCs using the RS232 serial port is purely because of the convenience of the implementation. A wider communication channel could be implemented if it is necessary. In this system, robots are purely effectors of agents. Agents can be seen as the "minds" of the robots. One robot can be represented by one agent, or many agents can manipulate one robot. Such flexibility is realised by the links between agents and robots. A robot exists in a form of an external function in ARBS, where an agent bears a robot's physical constraints and any of its actions can be carried out by calling the corresponding external function of the robot. If there are many agents who bear a robot's physical constraints and call the robot's function, then the robot is a single manipulator for many agents. This flexibility brings economic benefit for the test-bed and enables the author to investigate the optimal representation of an existing physical device.

The physical locations of two robots are illustrated in Figure 2.8, where two robots have an interaction area in order that some interactions can be performed. Despite the coordinate system used by each agent, there is an overall coordinate system used by the

system. The relationship of the different coordinate systems is also illustrated in the figure.

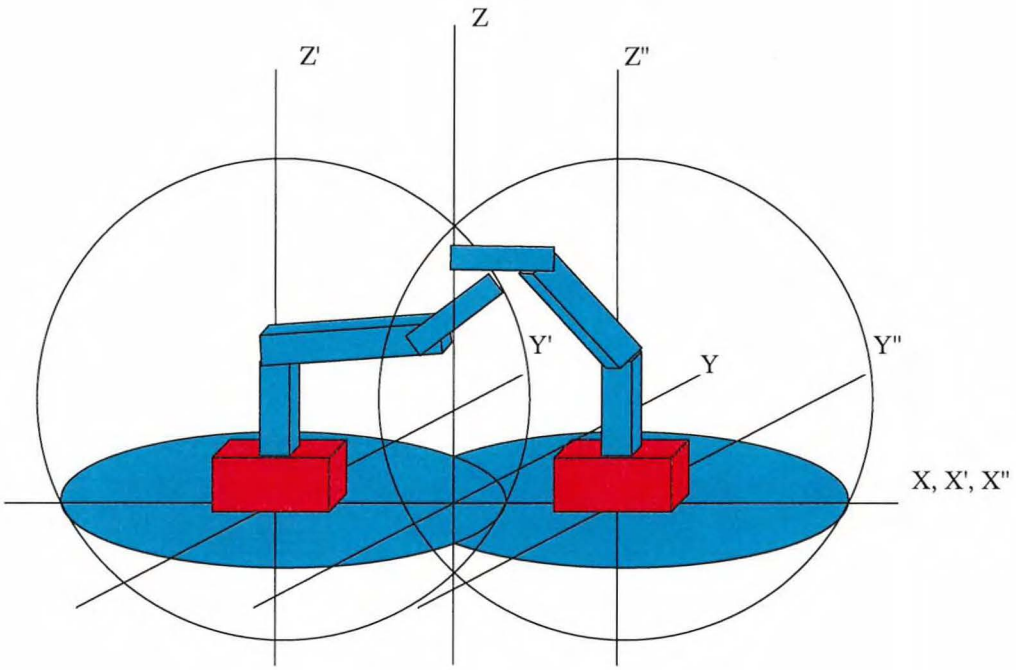


Figure 2.8 The locations of two robots and their coordinate systems

2.5 Summary

This chapter described a development of a multi-agent system. It is motivated by developing an applicable and simple-to-use framework for control of multiple robots' cooperation. This multi-agent system is not only for the application of multiple robots' cooperation but also for testing the effectiveness of existing cooperation schemes and identifying the theoretical requirements in development of a cooperation framework for multi-agent systems. In developing the system, ARBS, an in-house blackboard system, was adopted to construct a test-bed. The representation of individual agents was done by rules spread across the different KSs that represent the different steps in a task's performance, and a new paradigm for constructing and organising KSs was established. Using these a primary multi-agent system was constructed. Based on this multi-agent

system the control of two MA2000 robots' cooperation test-bed was constructed. This test-bed will be used for implementing and testing existing control schemes and enabling the identification of theoretical issues that need to be addressed.

Chapter 3

Implementation and Tests of Existing Frameworks(1) – The Contract Nets

Learning by doing --

No practice, no comments.

-- Mao Zedong (former president of China)

A huge elephant could be eaten by a family of tiny ants.

-- A Chinese proverb

Developing a new framework for multi-agent cooperation is a difficult task. At least two approaches can be taken to alleviate this problem. One approach is to build more multi-agent systems which can be classified as *learning by doing* and the other one is to draw upon ideas from other fields (e.g. management science, biology), which have considerable experience with their own multiple agents' systems. It can be classified as *learning by analogy*. The following two chapters will report the approach that adopts the *learning by doing* method. Empirical developments adopting influential cooperation frameworks were carried out to identify the shortcomings of the frameworks in the problem domain in order to establish the objectives for developing the new framework.

The process of developing a new multi-agent cooperation framework in this thesis can be viewed as a refinement of existing frameworks. Based on the test-bed constructed and described in the previous chapter, the two most influential multi-agent cooperation frameworks from the two research approaches identified in Chapter 1 were implemented. The refinement of these frameworks was revealed during the actual implementation and tests, where some problems had been solved by one framework and new problems were also perceived. Eventually the process of refining the frameworks presented in the following two chapters ends with objectives for the new framework.

The implementation and testing of the two existing cooperation frameworks are presented in a similar manner: a brief description of the cooperation framework is followed by the actual implementation on the test-bed, then tests based on generated hypotheses are provided, finally a discussion of the tests' results in which important points and perceived problems are presented as reasons for refinement.

In this chapter, section 3.1 provides a brief description of the first cooperation framework from the DPS approach namely the Contract Net framework. Section 3.2 provides detailed implementation of the framework on the test-bed. The tests and the results are reported in section 3.3, followed by a discussion of the results in section 3.4. Finally section 3.5 provides a summary to constitute reasons for the refinement.

3.1 The Contract Net Framework

Research in developing frameworks for cooperation in DPS has a long history and many cooperation frameworks have been developed [Durfee et al. 89, Smith and Davis 81, Findler and Gao 87, Kornfeld and Hewitt 81]. The Contract Net cooperation framework developed by Smith and Davis [Smith and Davis 81, Smith 80] is adopted and implemented because of two principal reasons. One is the function of the framework,

Smith and Davis state this as "developing a framework for cooperative behaviour between willing entities" [Smith and Davis 81]. This is exactly one of the objectives of this research. Another reason is its form of problem solving. Contract Nets were developed with the metaphor of a group of human experts working together trying to complete a large task. The cooperation is in the form of agents dividing the workload between them and each agent independently solving some subproblems of the overall problem. This metaphor is similar to the blackboard metaphor of ARBS.

The Contract Net framework was developed originally to solve the problem of allocating tasks in *task-sharing* cooperation [Smith and Davis 81]. Task sharing is a form of cooperation in which individual agents assist each other by sharing the computational load for the execution of subtasks of the overall problem.

In the Contract Net framework, a contract is an explicit agreement between an agent that generates the task (the manager) and an agent willing to execute the task (the contractor). The manager is responsible for monitoring the execution of the task and processing the results of the executions. The contractor is responsible for the actual execution of the task. The individual agents are not designated *a priori* as manager or contractor; these are only roles, and any agent can take on either role dynamically during problem solving. Agents are therefore not statically tied to a control hierarchy.

A contract is established by a process of local mutual selection based on a two-way transfer of information. In brief, the managers advertise the existence of the tasks to other agents with messages of the task announcement (Figure 3.1a). Available agents who are potential contractors evaluate task announcements made by the task managers (Figure 3.1b) and submit bids on those for which they are suited (Figure 3.1c). An individual manager evaluates the bids and awards contracts for execution of the task to the agents it determines to be the most appropriate (Figure 3.1d). Manager and

contractor are thus linked by a contract (Figure 3.1e) and communicate privately while the contract is being executed.

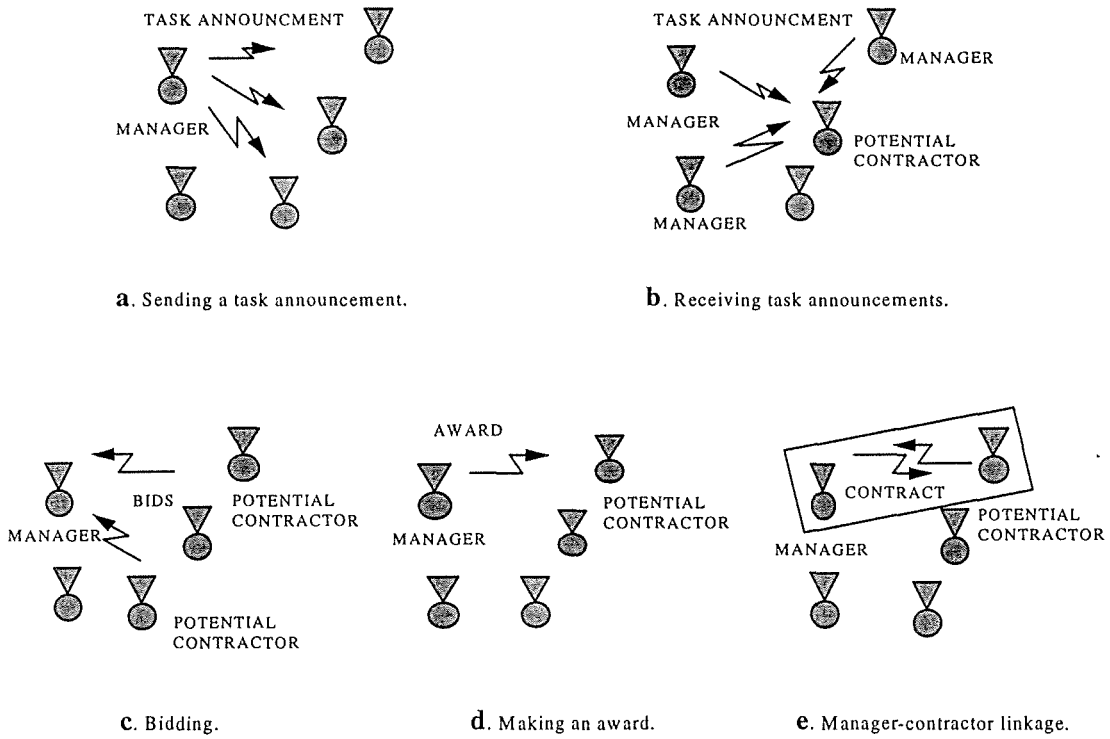


Figure 3.1 The Contract Net framework [Smith and Davis 81]

The negotiation process may recur. A contractor may further partition a task and award contracts to other agents. It is then the manager for those contracts. This leads to a hierarchical control structure that is typical of task sharing. The contents of the communication between manager and contractor are mostly that the manager supplies task information and the contractor reports progress and the eventual result of the task.

3.2 Implementation of the Contract Net Framework

Implementing the Contract Net framework in control of multiple robots' cooperation was straightforward once the test-bed had been constructed. However, two points associated with the implementation have to be clarified here. Firstly, the purpose of

implementing the Contract Net framework is to test its applicability and the possible inadequacy in the problem domain. It is not intended to provide a general judgement about the Contract Net framework itself. Therefore in order to anchor the framework into a context, a detailed description of the implementation on the test-bed is provided. It includes the blackboard partitioning, the messages used in the communication process, and the KSs' arrangement. Secondly the test, which will be described in the next section, and the results of these tests are based on the system that implements the Contract Net framework. This is to say that the tests' results and the discussion of the results are focused on the implemented system. They are also not a general judgement of the Contract Net framework. Therefore any claims made about the Contract Net framework by default only applies in the problem domain and the system built on the test-bed.

With this clarification, implementation of the Contract Net framework can be pursued. The implementation took place by re-configuring the general multi-agent system on the test-bed described in the previous chapter. It includes resetting the blackboard panels[1] and adjusting the rules and KSs to enable the Contract Net framework to take into account such factors as the number of agents that represent available robots; the number of robots available in the system; the communication methods; the message format and the working process of the cooperation framework.

3.2.1 Blackboard Partition

In the Contract Net framework, a contract is set up by a series of communications between a task manager and contractors. The communication process consists of *advertising*, *bidding* and *awarding*. This two-way information transfer involves formats

[1] Resetting the blackboard panels is realised by adding new sublists into the list of the blackboard list. It is an advantageous feature of ARBS (see section 2.1.4 in Chapter 2).

of one-to-one and one-to-many. A task manager who initiates a contract advertises the existence of the task to other potential contractors with a *task announcement message* (see Figure 3.3). The task announcement can be addressed to all agents in the system using general broadcast, which is *one-to-many* or to a single agent, which is *one-to-one*. Bidding is when a willing agent submits a bid in response to a particular task announcement. Awarding is where a task manager informs associated bidders that they are now contractors for the task. To support these different formats of communication the blackboard has to be partitioned with necessary message pools, which are the new panels in the blackboard. Extending the original blackboard partition strategy as a public information memory, this implementation sets up agents' private panels on the blackboard as can be seen in Figure 3.2 compared with Figure 2.5.

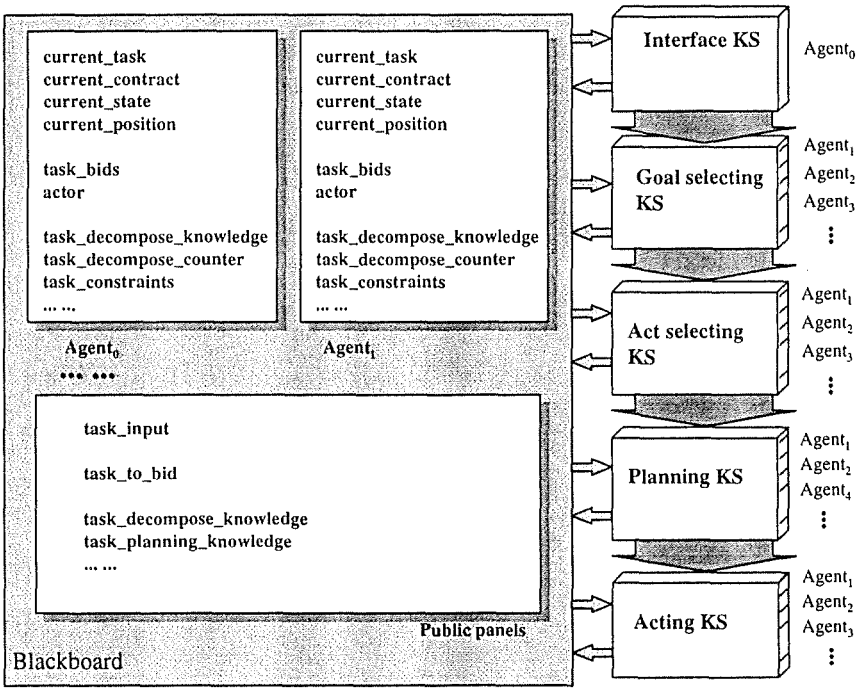


Figure 3.2 The blackboard partition for implement the Contract Net framework

The private panel strategy means that only the agent itself and the other agents who have permission from the agent can access the partitions in the private panels. For example, the current_task partition on $agent_i$'s panel can be accessed by $agent_i$ for checking if

there are still any tasks that need to be performed. It may also be accessed by $agent_2$ for awarding $task_7$ that $agent_1$ was bidding to perform. In this case $agent_2$ is the manager of $task_7$ and $agent_1$ gives permission to $agent_2$ to access its partition in the private panel with its bidding message. This is explained in more detail in the next section.

The contents of each partition in both agents' private and public panels are indicated by the name of the partitions. For example, current_task records the locally queued tasks that are waiting to be performed. Current_contract records the valid contracts with other agents in which both the name of the task manager and the contractors are recorded. Task_bids records the received bids of the advertised tasks. Task_decompose_counter records the subtasks decomposed from a task, with this record the agent is able to tell that a task is completed when all its subtasks have been completed. If there is a strict sequence required among the subtasks of a task, this is recorded in the partition task_decompose_counter. The partition task_decompose_knowledge records the agent's inherited and learnt knowledge. The actor partition is used to keep track of which robot is acting for which agent or agents. As described in section 2.4.1, the test-bed constructed in this research separates robots from agents. Many agents can represent one robot. The representation of a robot by an agent is realised with an external function, which identifies the functionality, the area of the robot's coverage and the physical constraints of the robot. The actor partition in an agent's panel keeps the functions name of a particular robot. In this way, the agent represents a robot with knowledge of the robots functionality, coverage and constraints, which will be used to select a task on which to submit a bid.

The task_decompose_knowledge and the task_plan_knowledge partitions in the public panel are used to store organisational knowledge [Steers 77]. This is the knowledge of a contract. After a contract has been successfully completed, all the agents in the system

learn the knowledge for future contracts.

3.2.2 Message Structure and Transfer

The Contract Net framework requires each message type to have slots for task-specific information. The slots have been chosen to capture the types of information that are usefully passed between agents to determine appropriate connections without excessive communication. In this implementation, three main types of message are shown in Figure 3.3.

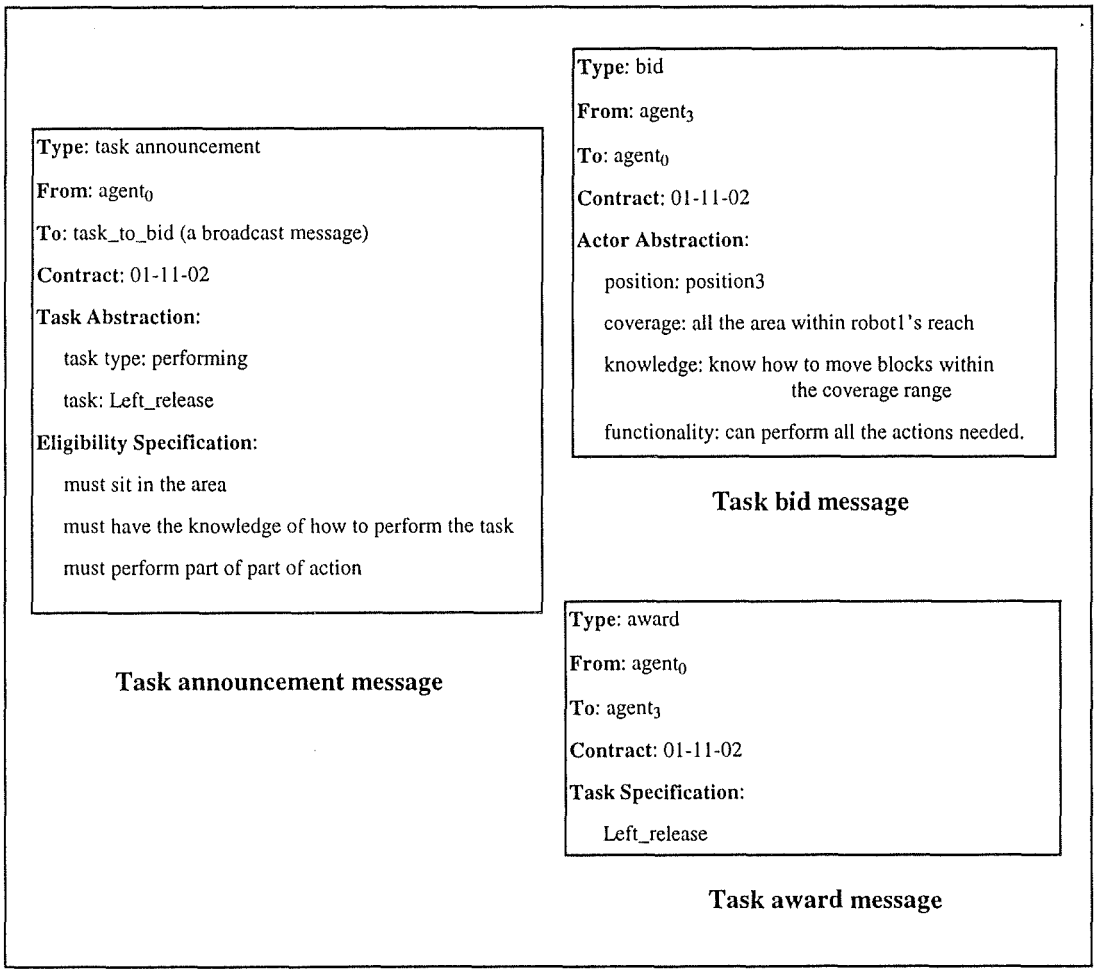


Figure 3.3 The structure of the messages used with the Contract Net framework

The way that an appropriate resolution is accomplished in this application was done with the powerful facility of the conclusion part of a rule provided by ARBS. In this a rule can add appropriate information onto a selected partition of the blackboard. For

example, if considering the message exchanged between a task manager and potential contractors, the task manager advertises its task using a task announcement message shown in Figure 3.3 and posts on the partition of task_to_bid on the public panel. This is a broadcast that every agent in the system is able to view. In the message, the *task abstraction* is both the type of task and the task itself. The task in the task slot of the task announcement message enables a potential contractor to determine to which one it should respond. The *eligibility specification* indicates that the only agents who should bid for this task are those which 1) must sit in the area, 2) have the knowledge of how to perform the task and 3) must perform part of the actions. This helps to reduce the chance of an over diligent agent detracting from the task performance efficiency by over bidding. If the task is a task that can be further decomposed into another series of subtasks, this eligibility specification requests that the bidder have the knowledge of further decomposition and can perform part of the actions. This means that the original task manager does not allow an agent who can only decompose the task to be eligible.

Each potential contractor watches task advertisements made by task managers. It checks its associated *actor abstraction* to see whether it is eligible and submits a bid (see task bid message in Figure 3.3). The bid submission is done by adding a task bid message to the task_bids partition on the task manager's panel. The task bid message supplies the name of the contractor, the contract number and the actor's abstraction. The manager uses this information to select a contractor and awards a contract (see task award message in Figure 3.3) by adding the task specification to the current_task partition and the task award message to the current_contract partition on the contractor's panel.

3.2.3 KSs and Their Order

In the test-bed constructed, the process of a task's performance is represented by KSs (see section 2.3). Comparing the existing task performance process in the constructed

test-bed (Figure 2.5) with the Contract Net framework, the following changes were made to enable the Contract Net framework to be implemented.

1. Change *interface KS* into advertising KS, which only consists of one agent that is the *general manager*[2]. This is similar to the task manager in the distributed sensing system [Smith and Davis 78], which Smith and Davis used to demonstrate the Contract Net framework. Both of them do not have task performance capabilities but do have extensive processing capabilities. They attempt to find agents to provide them with task performance.

1. Change *goal selecting KS* and *act selecting KS* into bidding KS. These are agents watching and submitting bids to the task manager.

2. Change *planning* KS into awarding KS.

3. Change the agents' rules in *acting* KS to enable the decomposition of combined tasks and advertising of decomposed subtasks to be carried out. Also change the action slot of the KS to activate the bidding KS.

Therefore the KSs and the order of the KSs in this implementation of the Contract Net framework are as illustrated in Figure 3.4.

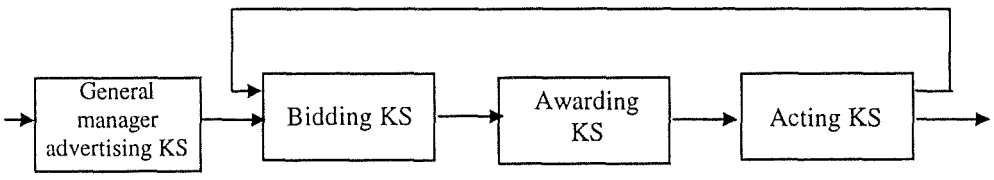


Figure 3.4 The KSs and their order in implement the Contract Net framework

The above KSs and the activating order of KSs show how the system works. When tasks have been inputted to the system, they all appear on the public panel of task_input. The

[2] Here the term *general manager* is used to distinguish the interface agent from other agents who can be a manager of a subtask.

first KS to be activated is the general manager advertising KS. It then generates a tasks' broadcasting message and broadcasts them on to the task_to_bid partition. This will trigger the bidding KS to become active and submit bids by adding them on to the manager's task-bids partition. After bids are received and all the rules in the bidding KS are exhausted, the awarding KS becomes active and awards the contract to the suitable contractors. The next KS to become active is the acting KS. All the agents represented by rules in this KS will perform the tasks they have bid for and have been awarded. The performances will differ according to the tasks. If a task, which has been awarded to an agent, can be directly performed, the agent will call its actor to perform the task and report the result to the manager. If a task awarded to an agent needs further decomposition then the agent will decompose it and post the decomposed subtasks on to the public panel of task_to_bid and activate the bidding KS to start another contract setting process.

3.3 Tests and Results

To test the applicability of the Contract Net framework in multiple robots' cooperation, the following hypotheses are generated:

1. The system should be able to perform all types of tasks that a multiple robots system should do.

2. By evenly distributing workload and sharing computational load and task performance, the system should achieve a relatively higher overall efficiency.

3. By distributing control and task performance, the system should result in a better fault tolerance and graceful degradation.

3.3.1 Testing Samples

To test the above hypotheses, a number of tasks are needed. Ideally, the tasks should be selected from the practice where multiple robots are involved. However, due to the constraints of available robots and the nature of the tests, the tasks used in the tests were artificially generated. The generation of tasks is based on the ability of available robots and the purpose of testing. Three types of task are generated as testing samples. They are:

1. *Atomic tasks*: these are tasks that the robots are designed for. For example *open* and *close* gripper, *rise up* and *lower down* the elbow and *roll* the "hand" which is called the 'end effector' in the MA2000 robot. Any MA2000 robot can perform these tasks and any other tasks are eventually performed by a combination of these tasks.

2. *Combined tasks*: these are tasks that can not be performed by any one robot with one act. They either need multiple robots to perform together where each robot performs a part of the task, or need one robot to perform a series of subtasks of the task. In other words, these tasks are combination of many atomic tasks. So they can be decomposed into a list of atomic tasks. Decomposition of a combined task is dependant on the nature of the task. No matter which agent carries out the decomposition process the result is the same list of atomic tasks. For example, task 'left_grip' (see Table 3.2) can be decomposed into two atomic tasks: move to the 'left' and 'grip'. This is a combination of one robot's task. Another example is 'hand_over', which can be decomposed into 'hand_out' and 'hand_in'. This needs two robots to perform it.

3. *Complex tasks*: the concept of complex tasks is introduced by this research to describe a task that cannot be simply decomposed into a list of subtasks or the task decomposition has several different results. For example, moving a large[3] object from one position to another position which can not be decomposed into a list of subtasks as the large object can not be taken apart. Another example is the well-known classical AI problem, *Tower of Hanoi*. There are different ways to achieve the accomplishment of this task depending on the problem-solving strategy adopted by the agent.

In total fifty tasks of the above three types have been generated. Table 3.1 lists twenty atomic tasks. They are represented in 'Posture Coordinates'. This means that performing a task is done by driving a robot to the required posture coordinates since the robot available does not have visual and sensor devices. Table 3.2 lists twenty combined tasks. They are represented with their component atomic tasks in order. The components of a combined task can be performed by one or two robots. Table 3.3 lists ten complex tasks.

[3] Here the large object means the object can not be moved by a single robot.

Table 3.1 Atomic tasks used for the test

Name	Posture Coordinates						
	Waist	Shoulder	Elbow	Pitch	Yaw	Roll	Griper
Park(1,2)	500	400	100	500	500	500	0
Bend_over	500	900	500	999	500	500	0
Lay_down	500	135	500	500	500	500	0
Sit_down	500	135	140	500	500	500	0
Grip	-*	-	-	-	-	-	1
Release	-	-	-	-	-	-	0
Gripper_up_down	-	-	-	-	-	999	0
Gripper_sideway	-	-	-	-	-	500	0
Left	150	900	500	500	500	500	0
Right	150	190	500	500	500	500	0
Back	500	90	500	900	500	500	0
Front	500	900	500	0	500	500	0
Inter_left	150	400	100	500	500	500	0
Inter_right	850	500	500	500	500	500	0
Inter_back	500	600	900	500	500	500	0
Inter_front	500	500	500	0	500	500	0
Position1	200	760	250	300	500	500	0
Position2	520	460	750	500	500	500	0
Position3	520	634	389	500	500	500	0
Position4	200	760	250	300	500	500	0

* - represents current coordinates

Table 3.2 Combined tasks used for the test

Task Name	Components
Hide	Sit_down, Bend_over, Lay_down
Lay_on _left	Left, Lay_down
Lay_on_right	Right, Lay_down
Lay_on_back	Back, Lay_down
Lay_on_front	Front, Lay_down
Sit_left	Left, Sit_down
Sit_right	Right, Sit_down

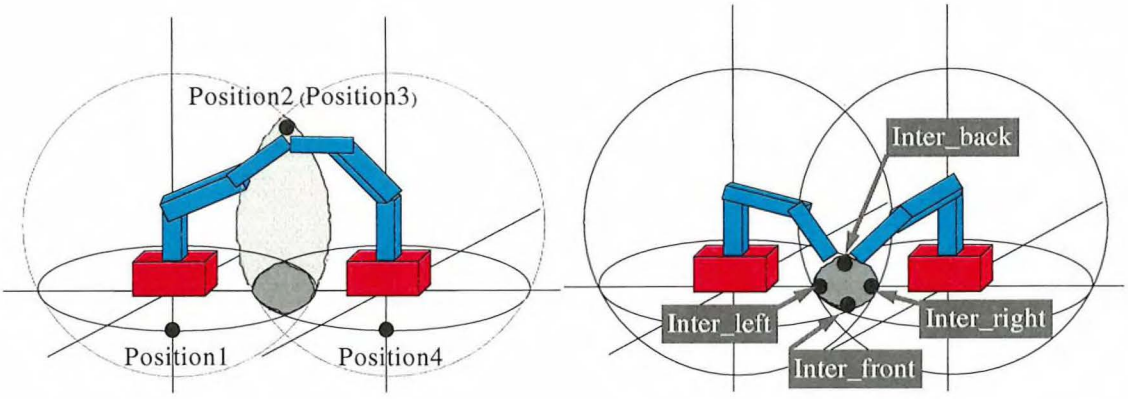
Sit_back	Back, Sit_down
Sit_forward	Front, Sit_down
Left_grip	Left, Grip
Left_release	Left, Release
Right_grip	Right, Grip
Right_release	Right, Release
Back_grip	Back, Grip
Back_release	Back, Release
Front_grip	Front, Grip
Front_release	Front, Release
Hand_over	Hand_out, Hand_in
Hand_out	Position1, Position2
Hand_in	Position3, Park1, Position4, Park2

Table 3.3 Complex tasks used for the test

Task Name	Description	Tasks
Move (initial, destination)	This task can be completed with different performance sequences. Solution1: initial, position2, position3, destination, park. Solution2: initial, position3, position2, destination, park.	Move(front, back), Move(left, right), Move(left, back), Move(front, right), Move(position1, position4),
Transport (initial, destination)	This task is assumed that a large object is needed to be moved. Two robots are needed to move the object in the same time.	Transport(inter_front, inter_back), Transport(inter_left, inter_back), Transport(inter_front, inter_right), Transport(inter_left, inter_right).
Hanoi_tower (pegs, disks)	This is a tradition AI task. The task itself may be varied with different numbers of pegs and different numbers of disks. There are two problem-solving algorithms which can be adopted by an agent. One is the least number of steps and the other is an algorithm that the agent has the most chance to participate in the performance[4].	Hanoi_tower(3, 5). The format of solution is move(peg, from, to).

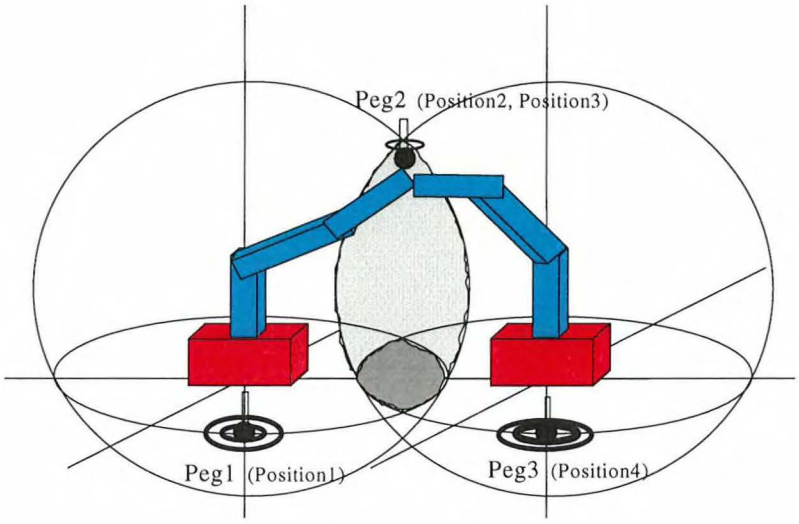
[4] In this algorithm, the adopting agent always chooses the movement that it can be involved in. For example, moving disk3 from peg1 to peg3, if the agent is sitting in peg2, it will adopt a plan that moves disk3 to peg2 and then moves from peg2 to peg3.

Task 'move' represents tasks that although they can be decomposed into a list of atomic tasks, the decomposition is not unique. In this particular testing sample, two different ways of decomposing a task were identified for the purpose of illustration. One is moving to position2 and then position3, which can be interpreted as 'loaded robot moves to pass-over position first'; another one is moving to position3 first and then position2, which can be interpreted as 'unloaded robot moves to pass-over position first'. Position2 and position3 are the same point in the overall system coordinates, but are different coordinates in the two robots' own coordinate systems. This has been shown in Figure 3.5(a). Both solutions can achieve completion of the tasks. The task 'transport' represents the tasks involved to move a 'large object' from an initial position to a destination. This particular task requires that the initial position and the destination are both located in the interaction area of the two robots. This is illustrated in Figure 3.5(b). The task 'Hanoi_tower' represents a task where the accomplishment of the task may be achieved by totally different performance plans. The difference includes the number of steps taken to complete the task and the actions in each step. This particular task generated here assumes that peg1 is located in position1, peg2 is located in position2 or position3 in robot2's coordinates, and peg3 is located in position4. This is illustrated in Figure 3.5(c).



a. Move (position1, position4)

b. Transport(initial, destination)



c. Hanoi_tower(3, 5)

Figure 3.5 Illustration of complex tasks

3.3.2 System Configuration

The system implementing the Contract Net framework had been configured with seven agents. One agent is used as the general manager and the remaining agents are executing agents. The reason for having so many agents is to investigate load distribution. The actors of the agents, their in-built knowledge, and their problem-solving strategy have been listed in Table 3.4.

3.3.3 Test Results

Fifty generated tasks were continuously input into the system with the above configuration. The results are summarised in Table 3.5.

Table 3.4 Configuration of the system used to test Contract Net

Name	Performing ability	Decomposing knowledge	Problem-solving strategy	Bidding strategy	Rewarding strategy	Actor
General manager	No.	Hand_over, Hide.	No.	No.	The first bidding received.	No
Agent1	All atomic tasks except Park2, Right, Back, Position3 and Position4[5].	Lay_on _left, Lay_on_right, Lay_on_back, Lay_on_forward.	No.	All eligible tasks' announcement.	The first bidding received.	Robot1
Agent2	All atomic tasks except Park2, Right, Back, Position3 and Position4.	Sit_left, Sit_right, Sit_back, Sit_foward, Hand_out.	Transport(initial, destination).	All eligible tasks' announcement.	The first bidding received.	Robot1
Agent3	All atomic tasks except Park2, Right, Back, Position3 and Position4.	Solution1 of Move (initial, destination).	Least performing steps for Hanoi_tower(3, 5).	All eligible tasks' announcement.	The first bidding received.	Robot1
Agent4	All atomic tasks except Park1, Left, Front, Position1 and Position2.	Left_grip, Left_release, Right_grip, Right_release.	No.	All eligible tasks' announcement.	The first bidding received.	Robot2
Agent5	All atomic tasks except Park1, Left, Front, Position1 and Position2.	Back_grip, Back_release, Front_grip, Front_release, Hand_in.	Transport(initial, destination).	All eligible tasks' announcement.	The first bidding received.	Robot2
Agent6	All atomic tasks except Park1, Left, Front, Position1 and Position2.	Solution2 of Move (initial, destination).	Favour of own performance for Hanoi _tower(3, 5).	All eligible tasks' announcement.	The first bidding received.	Robot2

[5] Ideally, all the atomic tasks should be performed by every agent. However, in order to demonstrate the tasks that need two robots to perform them, it is set up deliberately that some atomic tasks can not be performed by certain agents.

Table 3.5 The results of tests

Atomic Tasks	Status and performer	Combined Tasks	Status and performer	Complex Tasks	Status and performer
Park1	Done. Robot1.	Hide	Done. Robot1.	Move(front, back)	Done. Robot1, Robot2.
Bend_over	Done. Robot1.	Lay_on _left	Done. Robot1.	Move(left, right)	Done. Robot1, Robot2.
Lay_down	Done. Robot1.	Lay_on_right	Done. Robot1, Robot2.	Move(left, back)	Done. Robot1, Robot2.
Sit_down	Done. Robot1.	Lay_on_front	Done. Robot1.	Move(front, right)	Done. Robot1, Robot2.
Left	Done. Robot1.	Lay_on_back	Done. Robot1, Robot2.	Move(position1, position4)	Done. Robot1, Robot2.
Right	Done. Robot2.	Sit_left	Done. Robot1.	Transport(inter_front, inter_back)	Failed.
Front	Done. Robot1.	Sit_right	Done. Robot1, Robot2.		
Back	Done. Robot2.	Sit_front	Done. Robot1.	Transport(inter_front, inter-right)	Failed.
Grip	Done. Robot1.	Sit_back	Done. Robot1, Robot2.		
Release	Done. Robot1.	Left_grip	Done. Robot1.	Transport(inter_left, inter_back)	Failed.
Gripper_up_down	Done. Robot1.	Left_release	Done. Robot1.		
Gripper_sideway	Done. Robot1.	Right_grip	Done. Robot2, robot1.	Transport(inter_left, inter_right)	Failed.
Inter_left	Done. Robot1.	Right_release	Done. Robot2, robot1.		
Inter_right	Done. Robot2.	Front_grip	Done. Robot1.	Hanoi_tower(3, 5)	Failed.
Inter_front	Failed	Front_release	Done. Robot1.		
Inter_back	Done. Robot2.	Back_grip	Done. Robot1, Robot2.		
Position1	Done. Robot1.	Back_release	Done. Robot1, Robot2.		
Position2	Done. Robot1.	Hand_over	Done. Robot1, Robot2.		
Position3	Done. Robot2.	Hand_out	Done. Robot1, Robot2.		
Position4	Done. Robot2.	Hand_in	Done. Robot1, Robot2.		

In Table 3.5, there are a few tasks which the system failed to complete. The actual scenarios of the failure are provided here for a further analysis. The first failure of performance was the task 'inter_front'. This was a surprise since the task is an atomic task in nature and it should be performed without any problem. The actual problem which occurred was when robot1 moved from its previous position inter_left to its new position inter_front, while robot2 was located in position inter_right. See Figure 3.6.

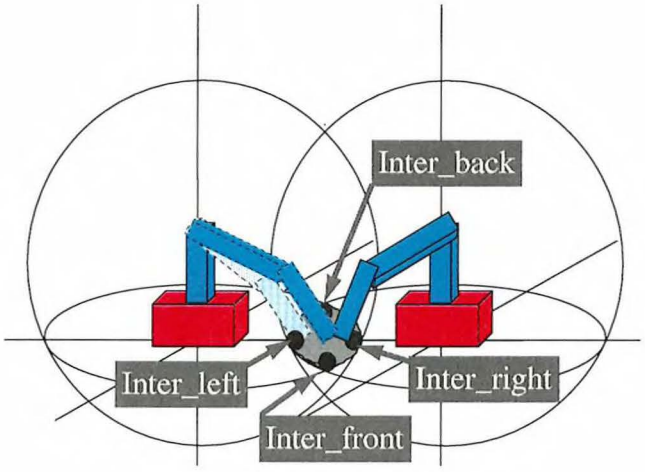


Figure 3.6 Robot2 blocks the path of robot1

In order to move to its new position, robot1 firstly raises its forearm and then runs into the forearm of robot2. This causes an emergency stop in the system and to stop performing any other tasks. Although it appears to be an accidental event, it exposes an important inadequacy of the system. This is that when the agents in the system lack an ability to foresee a potential conflict with other agents, there is no mechanism to prevent this happening and to issue any necessary rescue methods. The second failure was whilst performing the complex task 'transport'. This is not as surprising since the agent who provides the solution to the task only specifies the actions necessary to complete the task, but did not provide any coordinating method. Therefore, the result was that the object was not moved. This exposes an interesting problem in that the system is inadequate to complete tasks which can not be decomposed into atomic tasks. The last

failure was whilst performing the complex task 'Hanoi_tower (3, 5)'. Although the disparity between different solutions for performing the task were solved by the manager's rewarding policy that is 'the first bid wins the task', the subtasks were neither atomic tasks nor combined tasks. Therefore no agents in the system bid to perform them[6]. It reveals an important task format, that is the subtasks of the task are not performable. Therefore the overall task remains unperformed.

The above three occasions expose the shortcomings of the system. Nevertheless, they also reflect the Contract Net framework in this particular application domain. A further discussion of the Contract Net framework in the problem domain is provided in the next section.

3.4 Discussion of the Contract Net Framework

Discussion of the Contract Net framework is focused on the testing hypotheses stated at the beginning of section 3.3 taking the tasks that failed to be performed by the system implementing the Contract Net framework.

3.4.1 Applicability

The test results expose some failures in performing certain tasks. The reasons for those failures are not merely due to the implementation. The implemented framework shares the responsibility with its inherited inadequacy. The Contract Net framework assumes any task can be decomposed sufficiently into subtasks and the decomposed subtasks can be performed by agents working independently. The tests expose that it is not always the case for some tasks. The task 'Hanoi_tower' is an example of a task that cannot be decomposed into a list of subtasks that can be performed by individual agents

[6] The problem solving strategy adopted by different agents ends up with a solution in a form of 'move(peg, from, to). This format is not an atomic task or combined task.

independently. The task 'transport' exposes another kind of task where the decomposed subtasks need to be performed in a coordinated way. The results of the tests demonstrate the inadequacy of the framework in performing these two particular types of tasks.

3.4.2 Coherent Behaviour Assurance

Another problem arises from the tests. The Contract Net framework assumes that any contract can be successfully executed. This was not always the case with these tests. For example, an atomic task could fail to be performed (e.g. task 'inter_front'). A similar failure could arise when performing any task where the atomic task is a subtask of the task. The problem in the Contract Net framework is that there is no mechanism to ensure the actual processing of the subproblems. Here the actual processing of the subproblems includes both success and failure. The point is that if things go wrong, or a contractor perceives something will go wrong, the contractor should report to the manager. In this case a recovering action can be taken to resolve the problem. The problem is that there are no formal obligations or conventions associated with a contract in the Contract Net framework.

3.4.3 Application Dependency

In the system, the choice of strategy whereby a contractor chooses a task manager to submit a bid and whereby a task manager selects successful contractors to award a contract is application dependent. This is not a system specific feature. It also appears in the distributed sensing system, where Smith and Davis demonstrate the Contract Net [Smith and Davis 78], in which the contractor adopts the strategy of responding to the nearest manager. The manager also selects the nearest suitable contractor to issue its rewards. In the system, which implements the Contract Net framework in multi-robot domain, the contractor submits a bid to all managers and the manager awards a contract

to the agent from which the first bid is received. The experimental results show that with these two strategies the workload of the tasks' performance to each agent is dependent upon the inference engine and the logic order of the rules in KSs. In the tests, the problem is that the logically prior rule (i.e., agent1 and agent4) will have more chances to win a contract. This can be seen from the tests' result where if two robots are able to perform the same task, the task is always performed by robot1. If agents have the same knowledge of performing a task, the winner of the bidding is always the agent who is logically prior. Therefore the advantage of the Contract Net framework that it evens workload distribution is not an advantage in these tests. These two examples demonstrate that the strategy adopted by the task manager and contractor is application dependent. It means that the system designer has to carefully design and engineer the strategy before the Contract Net can be implemented. Such design and engineering needs to be done in such detail that there is little difference from the traditional top-down system design, which is against the initial objectives of this research.

3.4.4 Control Centralisation

The initial motivation of the Contract Net framework is to distribute task performance as well as control to achieve an extensive cooperation. It had been realised in these tests that agents have no pre-defined rules. Any agent can be a contractor and manager for different subtasks. However, the Contract Net framework does require a task manager to have knowledge about the task in order to decide how to describe the task and how to generate a potential contractor's eligibility specification. In the tests all tasks to the system were first generated by the *general manager* and the general manager announced the tasks to the public. It requests that the general manager describe each tasks announcement in substantial detail. All the other distribution including control and performance was based on this root of a hierarchy structure. Therefore if the general

manager fails to describe a task in an appropriate way the task will not be performed because no agent will bid for the task's performance. This means that a tasks success is dependent on the correct behaviour of the general manager. Its systemic behaviour becomes similar to the traditional centralised DPS systems.

3.5 Summary

The Contract Net framework offers a powerful mechanism for structuring high-level interactions between agents for a cooperative performance of tasks. It stresses two-way information transferral and mutual selection as the interaction mechanism of agents. This enables task distribution with distributed control and shared responsibility to maintain systemic reliability and communication efficiency. The implementation of the Contract Net framework in the control of cooperation between multiple robots on the test-bed confirms the benefits claimed by the framework. These include extensibility, and the ability to perform combined tasks.

However, the implementation also indicates the problems that the Contract Net framework has in the control of multiple robots' cooperation. The first problem is its applicability. It does not suit the performance of tasks that cannot be decomposed into a list of subtasks or where the decomposed subtasks are not atomic tasks. The second problem is the lack of a mechanism in the framework to ensure the responsibility of a contractor to its manager. It is an unsolved problem in the Contract Net framework as to how to achieve coherent and social behaviour in a system where the control is distributed among a number of autonomous agents. The third problem is its application dependency. This means that the performance of the system depends greatly on the strategy adopted by the system designer for a contractor to submit a bid and for the manager to award a contract. The last problem is the role of the general manager in the

system. It has little difference to a central controller in a conventional multiple robots systems, which damages the distribution of control and KSs in performing subtasks of a task. No matter how proportionate the control and KSs are distributed, if the general manager fails to perform the whole system fails. It is possible to delete the general manager in the application, but not to delete the role the general task manager performs. The broadcasting has to be done in a way that specifies each tasks announcement message including the eligibility specification and awarding contracts to contractors.

These existing problems with the Contract Net framework form the motivation for this research in seeking an extension, refinement and even a new cooperation framework.

Chapter 4

Implementation and Tests of Existing Frameworks(2) –The CPS Framework

Learning by doing --

Knowledge comes from practice.

-- Mao Zedong (former president of China)

A family of tiny ants could eat a huge elephant.

-- A Chinese proverb

The Contract Net cooperation framework, which is one of the most influential DPS approaches, has been examined and reported in the previous chapter. Now as the second part of the implementation and testing of existing frameworks, this chapter will report a study of the CPS approach in general. The focus is concentrated on one of the most influential frameworks in the CPS approaches proposed and formulised by Wooldridge and Jennings [Wooldridge and Jennings 94, 94(2)], namely the CPS framework.

This chapter is organised in a similar fashion to Chapter 3. Section 4.1 provides a brief description of the CPS framework. Section 4.2 describes the implementation of the CPS framework on the test-bed to form a system for testing. The tests and results are reported in section 4.3, followed by a discussion of the results in 4.4. Finally a summary

is provided in section 4.5 to facilitate the objectives of the new framework.

The CPS framework is adopted to study because:

- it may offer a means of resolving the problems perceived in the implementation of the Contract Net framework under DPS approaches, and
- it will also help to develop a framework for control of multiple robot cooperation.

The CPS approach is in contrast with DPS approaches such as Contract Nets. It is focused on how to achieve a cooperative and intelligent behaviour between a collection of autonomous intelligent agents [Bond 88]. Besides the well-known reasons for the usefulness of distributed systems in general, there are the following additional reasons for studying CPS approach.

1. Modularization. A multi-agent system under the CPS approach can be developed independently. It can also be reused as a component of a new larger multi-agent system. In the new larger multi-agent system the previously developed multi-agent system can be viewed as a member agent. This modularization can be very useful for developing a large or naturally distributed system.

2. Robustness. A system designed by the CPS approach can be more robust than otherwise, since the acquisition and validation of design requirements are simpler for such a system. The design of this system for a particular application can also be simpler by allowing an intelligent agent to be located at the site where the data are available and where the decisions have to be taken.

3. Flexibility. A system based on the CPS approach can distribute the expertise according to their natural distribution over agents who specialise in different

domains and dynamically team up the necessary agents to solve current problems.

Multi-agent systems based on the CPS approach have been the focus of great attention. A number of general frameworks have been developed. For example, Werner [Werner 89] developed a formal account for the design of systems in which agents behave as a social unit or group. Gasser and his colleagues [Gasser et al. 89] developed a framework for representing multi-agent systems based on a basic view that a cooperation framework is a particular set of settled and unsettled questions about belief and action that agents have about other agents. Bond [Bond 90] shares the same view with Gasser and his colleagues. He uses the concept of commitment introduced by Becker [Becker 60] in sociology. In Becker's description, individual agents participate in several organisations or settings. Hence, to regard their behaviour in any one setting as consistent lines of activity, the notion of commitment consequent to the individual's participation in other settings must be introduced. These commitments constrain the individual agent's action and can be used explicitly in negotiation between agents. Commitment appears then as a central concept of multi-agent systems. Jennings [Jennings 93] proposed a framework for multi-agent systems that agents should not only be constrained by commitments to the community and to themselves but also should be constrained by conventions that specify what a responsible agent should do when things went wrong. He argues that commitments and conventions are the foundation of coordination and cooperation in multi-agent systems. Unlike the contributions cited above, Wooldridge [Wooldridge 92, Wooldridge and Jennings 94, Wooldridge and Jennings 94(2)], based on Jennings' foundation of commitments and conventions, developed a general framework for multi-agent systems. He hoped that by constructing a formalism for multi-agent systems, which might be used in the specification and verification of realistic multi-agent systems, it could bridge the gap between the theory

of multi-agent systems and the practice of DAI.

In this research, Wooldridge and Jennings's CPS framework is adopted and implemented on the test-bed, firstly as a means of resolving the problems perceived in the implementation of the Contract Net framework and secondly to building a practical framework for multiple robot cooperation application. What is more important, it is hoped that by implementing the CPS framework a cross-fertilisation can occur between the application of managing and maintaining cooperative behaviours of robots and a formal modelling of cooperative multi-agent systems.

To sum up, the three objectives for implementing the CPS framework are:

- To resolve the problems perceived in implementing the Contract Net framework.
- To help develop a cooperation framework for multiple robots.
- To help structure a formal model for a cooperative multi-agent system.

4.1 The CPS Framework

The CPS framework [Wooldridge and Jennings 94(2)] is a four-stage mathematical model described in multi-modal logic. It serves as an abstract, top-level specification for building a CPS system. The framework is developed on the basis that the key mental states that control an agent's behaviour are intentions and joint intentions. The former defines an agent's local behaviour; the latter controls an agent's social behaviour [Bratman 87]. Intentions are important as they provide both stability and predictability through the notion of commitment [Becker 60], which is needed for social interactions, and the flexibility and reactivity that are required to deal with a changing environment. Intentions and joint intentions are described by commitment and its underlying convention [Jennings 93].

The four stages of the framework are [Wooldridge and Jennings 94(2)]:

1. **Recognition:** This is the beginning of a CPS process where some agents recognise the potential for cooperation. This recognition may come about because an agent has a goal that it is unable to achieve in isolation, or, more generally, because the agent prefers assistance.

2. **Team formation:** During this stage, the agent that recognised the potential for cooperative action at stage (1) solicits assistance. If this stage is successful, then it will end with a group having a joint commitment to collective action.

3. **Plan formation:** During this stage, the agents attempt to negotiate a joint plan that they believe will achieve the desired goal.

4. **Team action:** During this stage, the newly agreed plan of joint action is executed by the agents, which maintain a close-knit relationship throughout; this relationship is defined by an agreed social convention, which every agent follows.

This framework describes a model of CPS that is in evidence in most multi-agent systems [Wooldridge and Jennings 94(2)]. It has been viewed as a powerful framework with regards to its ability to manage cooperative actions among a group of agents in a system during run-time and to characterise various aspects associated with CPS over other frameworks [Durfee 88, Smith 80].

4.2 Implementation of the CPS Framework

To implement the CPS framework for control of multiple robot cooperation in the test-bed, the previous blackboard partition was still used. The main difference between the CPS framework and Contract Nets is that the CPS framework involves setting up joint commitments and corresponding conventions in a cooperative team action instead of a

contract that is set up by the Contract Nets. In the CPS framework, the commitments and conventions are represented by a set of rules that defines how an agent should behave when performing a team action under certain circumstances, such as a change to the environment. This set of rules needs to be represented in an appropriate degree of detail which potential team members can use to make a decision as to whether or not to join the team. However, there is no fundamental change in the format of the information exchange between the agents. The Contract Net framework can be still used as a communication protocol as can most of the blackboard partitions and the message structure in the implementation of the Contract Net framework. Minor changes were made to stress the differences of the CPS framework.

4.2.1 Blackboard Partition

The blackboard partition for implementing the CPS model is shown in Figure 4.1. Both public and private partitions are still used. Notice the difference between Figure 4.1 compared with Figure 3.2, which showed the blackboard partition for implementing the Contract Net framework.

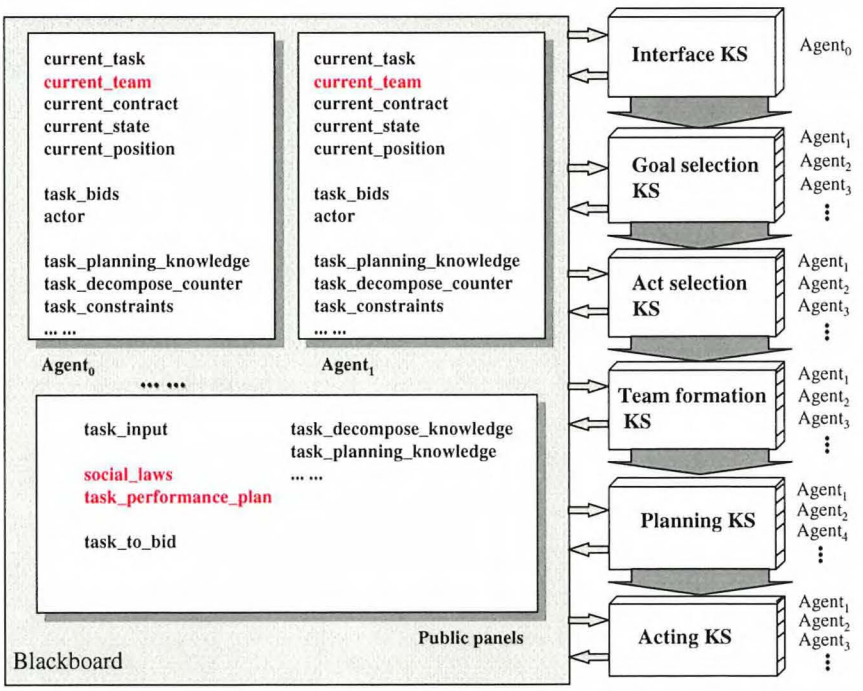


Figure 4.1 The blackboard partition for implement the CPS framework

There are two major differences compared with the Figure 3.2. The first one is the presence of two new public partitions called social_laws and task_performance_plan. The social_laws partition is used to store commitments and conventions associated with a team. Normally, when an agent recognises the need for cooperative actions in a task, it will post the task to the other agents. Because the cooperative action is performed in the form of team action in the CPS framework, it also needs to specify the team's identity at the same time. In this case it is a set of rules that potential team members should obey when they join the team. The reason why the social_law partition is in the public area is so that all agents can view it and can use it to make a decision as to whether or not to join the team. The task_performance_plan partition is used to store plan proposals by members of a team and the final plan that the team adopts as an executing plan for the tasks performance in the team. The second major difference between the two figures is a newly added agent's private panel called current_team. It is used to store information about the current team that an agent is engaged with. Using this information an agent knows its current team and the team members. When an unexpected event occurs, the agent knows to whom it should be obligated, what its obligations are and how to behave in such an unexpected event.

4.2.2 Message Structure and Transfer

The CPS framework also requires messages to capture the useful information passing between agents in different stages. There are mainly three rounds of information passing between agents. Each round has a different goal that shows the purpose of the message passing. These three rounds of message passing can be identified by their different goals, which are team establishment, plan formation and team action. In each message passing round three types of message are used. They are similar to the messages used in the implementation of the Contract Net framework.

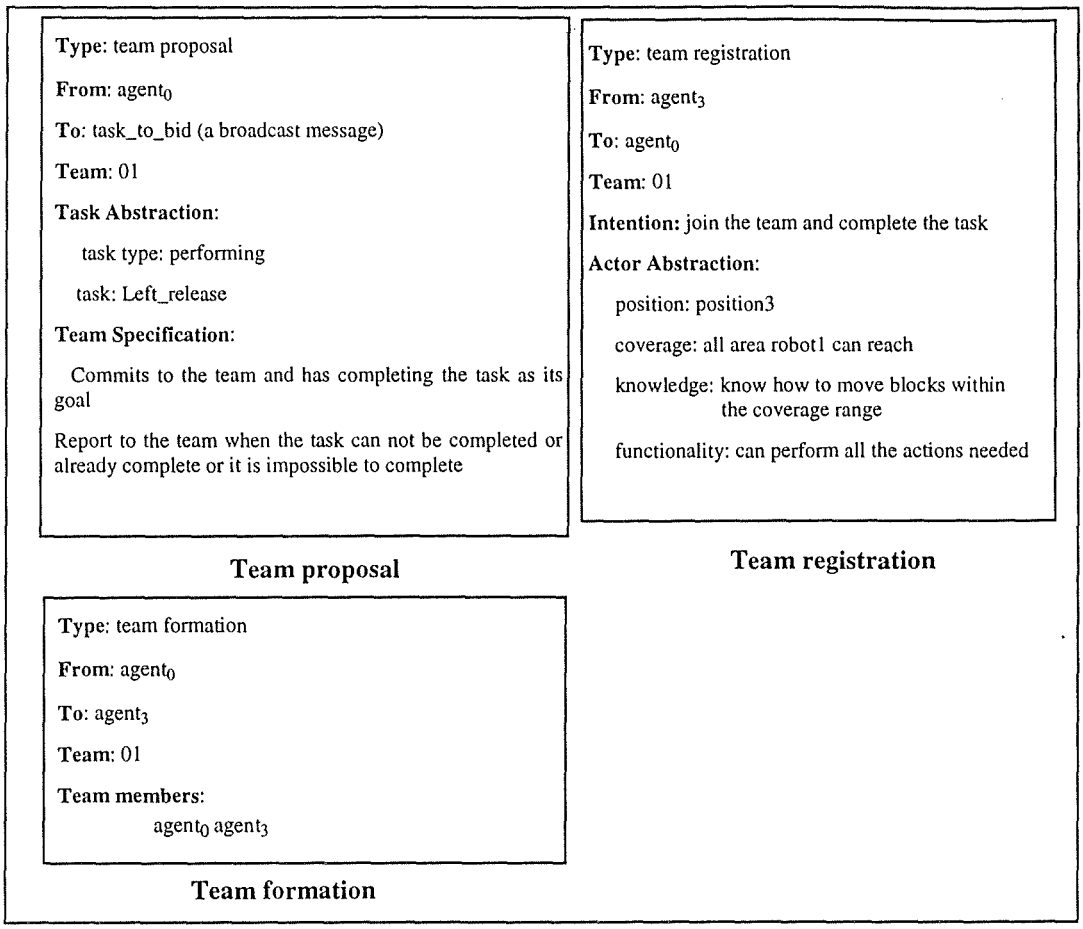


Figure 4.2 The structure of the messages used in the CPS framework

Figure 4.2 illustrates the messages used for the team establishment. The message transfer was performed by the rules in KSs provided by ARBS to add or remove information on a blackboard panel. In the team establishment process, an agent that recognises the potential for cooperation, either because it is unable to perform the task in isolation or because the agent prefers assistance, would propose a team formation using the team proposal message shown in Figure 4.2.

The team formation proposal explicitly indicates the team's goal, which is the task that needs to be accomplished and the rules under which the team operates. The actual proposal was performed by adding the task to the task_to_bid partition and the behaviour rules to the social_laws partition of the blackboard. This is similar to the task broadcast event in the Contract Net framework, where every agent in the system is able to view the team proposal. The potential team member checks its actor abstraction,

current tasks and the team rules to decide whether or not to respond with a team registration message to indicate that it is intending to join the team. The last step of the message passing in the team establishment is that the agent, which proposed the team formation, sends a Team formation message to all the team members to inform them of the establishment of the team and the members of the team (see Figure 4.2).

Similar message structures and information passing techniques were used in the other two rounds of message passing in the CPS model.

The second round of message passing is plan negotiation, which is a complex task [Fikes and Nilsson 71, Lansky 87, Pednault 87, Conry et al. 88]. It involves many complex issues such as specifying the negotiation language (e.g., STRIPS [Fikes and Nilsson 71]), the evaluation of plans and how to resolve the disparities between negotiators about a particular plan [Georgeff 83 and Stuart 87, Conry et al. 88]. Since the CPS model does not provide any particular planning mechanism, three simple planning mechanisms were identified for adoption:

1. **No planning**. No planning means that team members are not involved in any planning process. The team simply adopts a plan provided by the agent of the team proposer. It requires that the team proposer provide a plan about how to accomplish the task when proposing the team formation.

2. **First come first served.** In this planning scheme, all members in a team have a chance to provide a plan but the team will only adopt the first plan that has been proposed.

3. **Decided by the team proposer**. In this planning mechanism, every agent in a team has to supply a plan but it is up to the team proposer to decide which one to adopt. For example it may choose a plan with the fewest steps.

In any case, the plan is added to or removed from the task_performance_plan partition in the public panel of the blackboard using a simple message structure that has only three slots: the agent name, the task name and the plan. The final plan is stored in the task_performance_plan partition in the public panel of the blackboard.

The final round of message passing in the CPS framework is the team action. In this implementation the task distribution scheme, the message types and the methods of passing messages used in the Contract Net framework were adopted to allocate tasks to the team members. The task manager in this case is the initial team proposer. All the team members are the contractors.

4.2.3 KSs and Their Orders

To realise the CPS framework on the test-bed the KSs and the order of the KSs were arranged in the manner shown in Figure 4.3.

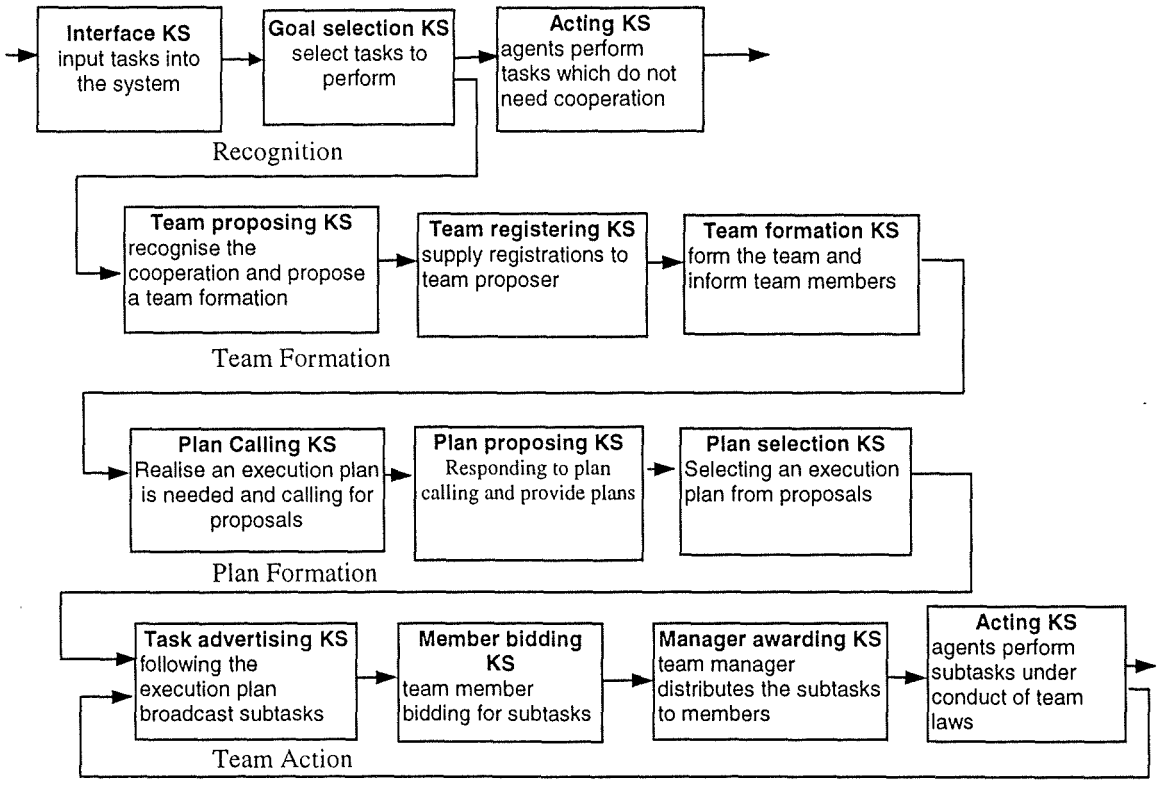


Figure 4.3 The KSs and their order in implementing CPS framework

In Figure 4.3, tasks inputted to the system initially appear on the public panel task_input. All the agents in the system are able to view the tasks. The agents in the system select tasks to perform according to their current states, position and abilities. These actions were realised by rules in the goal selecting KS and the acting KS. In the goal selecting KS, there are also rules, which represent the agent's recognition of cooperative action. The condition part of the rules is a list of conditions that represent the circumstances under which an agent recognises the need for a cooperative action, and the conclusion part of the rules is an action which will activate the team formation KSs if the rule is fired.

In team formation, the first KS activated is the team proposing KS. In this, rules, which represent the agents that recognise the cooperation, will generate a team formation message proposal and then broadcast the message to all potential team members. This will activate the team registering KS, in which all the agents, who have viewed the proposals for team formation, will respond by a team registration message or ignore the proposal. When all the rules in the team registration KS become exhausted, the team formation KS is activated so that the team proposer collects the agents' registrations and informs all the team members of the establishment of the team.

In plan formation, the three planning mechanisms described in the previous subsection were realised in the following manner. The initial team proposer has the right to be the team manager or choose another agent as the team manager. If the team proposer has a task execution plan, it will simply advertise subtasks according to the plan and ignore the plan formation stage, and go straight to the last stage of the CPS framework by activating the task advertising KS. This is slightly different from the original CPS framework. The reason for doing this is provided in discussion of the CPS framework, where the conditions of a cooperative action have been specified as being too narrow. If the initial team proposer does not have an execution plan, it will activate the plan

proposing KS, in which all the team members are required to supply a performance plan for a task. It is the team proposer's responsibility to select a plan to execute. In this implementation, either the "first come first served" mechanism or the "team proposer decides" mechanism was adopted. In either case, the termination of the plan selecting KS will activate the KSs in the last stage of the CPS framework. Here the previously developed system for implementing the Contract Net framework was reused. The task manager is the team proposer. It advertises all the subtasks in the selected execution plan to other team members. The order of KSs and the order of rules was the same as that used in the implementation of the Contract Net framework.

4.3 Tests and Results

To test the applicability of the CPS framework and the refinement of the Contract Net framework in multiple robots' cooperation, the following hypotheses were generated:

1. The system should be able to resolve the problems perceived in the implementation of the Contract Net framework and to perform all types of tasks that a multiple robots system should do.

2. As one of the initiatives of the CPS framework is to bridge the gap between theory and practice, implementing the CPS framework should not have many difficulties.

3. As another initiative of the CPS framework is that it will serve as a general framework of multi-agent cooperation, it should characterise various aspects associated with multi-agent cooperation.

4.3.1 System Configuration

With the above testing hypotheses, the system implementing the CPS framework had been configured with six agents. Compared with the agents used in the system

implementing the Contract Nets there was no agent acting as general manager. This is because the CPS framework requires the task to be posed to the system where every agent in the system can view it. The six agents in the system had the same functionality and task decomposing knowledge as the agents had in Contract Nets. Apart from this inheritance, the difference between these agents reflects the options provided by the CPS framework. They were the conditions that each agent adopts to propose a team formation, the commitments and conventions that team proposer sets for the team members, and the strategy that a team proposer adopts in choosing a plan to perform the task. The subtask bidding and rewarding strategy were the same as that used in the Contract Nets implementation. The configuration is listed in Table 4.1, where the options provided by the CPS framework were arbitrarily set to each agent.

4.3.2 Test Results

To test the CPS framework against the initial objectives of implementing the framework and the testing hypotheses stated in the beginning of the chapter, the same series of tasks that were generated for testing the Contract Net framework had been used on the CPS implementation. Here the overall testing results are listed in Table 4.2 to provide a basis for discussion. The details, difficulties experienced, and modifications made during the programming and testing are provided in the next section.

Table 4.1 Configuration of the system used to test CPS framework

Name	Conditions of team formation	Rules for its team members	Plan generate strategy	Bidding strategy	Rewarding strategy	Actor
Agent1	If it is unable to perform a task in isolation.	Commit to the team; report the success or failure of subtask's performance.	Adopts the first received plan.	All eligible tasks' announcement.	The first bidding received.	Robot1
Agent2	Prefers assistance.	Commit to the team; report the success or failure of subtask's performance.	Decided by itself.	All eligible tasks' announcement.	The first bidding received.	Robot1
Agent3	Prefers assistance	Commit to the team; report the success or failure of subtask's performance.	Decided by itself.	All eligible tasks' announcement.	The first bidding received.	Robot1
Agent4	Prefers assistance	Commit to the team; report the success or failure of subtask's performance.	Adopts the first received plan.	All eligible tasks' announcement.	The first bidding received.	Robot2
Agent5	If it is unable to perform a task in isolation.	Commit to the team; report the success or failure of subtask's performance.	Adopts the first received plan.	All eligible tasks' announcement.	The first bidding received.	Robot2
Agent6	If it is unable to perform a task in isolation.	Commit to the team; report the success or failure of subtask's performance.	Decided by itself.	All eligible tasks' announcement.	The first bidding received	Robot2

Table 4.2 The results of tests

Atomic Tasks	Status and performer	Combined Tasks	Status and performer	Complex Tasks	Status and performer
Park1	Done. Robot1	Hide	Done. Robot1.	Move(front, back)	Done. Robot1, Robot2.
Bend_over	Failed.	Lay_on _left	Done. Robot1.	Move(left, right)	Done. Robot1, Robot2.
Lay_down	Done. Robot1.	Lay_on_right	Done. Robot1, Robot2.	Move(left, back)	Done. Robot1, Robot2.
Sit_down	Done. Robot1.	Lay_on_front	Done. Robot1.	Move(front, right)	Done. Robot1, Robot2.
Left	Done. Robot1.	Lay_on_back	Done. Robot1, Robot2.	Move(position1, position4)	Done. Robot1, Robot2.
Right	Done. Robot2.	Sit_left	Done. Robot1.	Transport(inter_front, inter_back)	Done. Robot1, Robot2.
Front	Done. Robot1.	Sit_right	Done. Robot1, Robot2.		
Back	Done. Robot2.	Sit_front	Done. Robot1.	Transport(inter_front, inter-right)	Done. Robot1, Robot2.
Grip	Done. Robot1.	Sit_back	Done. Robot1, Robot2.		
Release	Done. Robot1.	Left_grip	Done. Robot1.	Transport(inter_left, inter_back)	Done. Robot1, Robot2.
Gripper_up_down	Done. Robot1.	Left_release	Done. Robot1.		
Gripper_sideway	Done. Robot1.	Right_grip	Done. Robot2, robot1.	Transport(inter_left, inter_right)	Done. Robot1, Robot2.
Inter_left	Done. Robot1.	Right_release	Done. Robot2, robot1.		
Inter_right	Done. Robot2.	Front_grip	Done. Robot1.	Hanoi_tower(3, 5)	Done. Robot1, Robot2.
Inter_front	Done. Robot1.	Front_release	Done. Robot1.		
Inter_back	Done. Robot1.	Back_grip	Done. Robot1, Robot2.		
Position1	Done. Robot1.	Back_release	Done. Robot1, Robot2.		
Position2	Done. Robot1.	Hand_over	Done. Robot1, Robot2.		
Position3	Done. Robot2.	Hand_out	Done. Robot1, Robot2.		
Position4	Done. Robot2.	Hand_in	Done. Robot1, Robot2.		

4.4 Discussions of the CPS Framework

Similar to the discussion provided in Chapter 3, the discussion is focused on the testing hypotheses stated in the beginning of the section 4.3.

4.4.1 Improvement

The three main problems associated with the Contract Net framework in the control of multiple robots' cooperation were 1) failure to perform some complex tasks, 2) general manager dependence, and 3) lack of assurance regarding coherent behaviour. These three problems are resolved by the system implementing the CPS framework. Considering the first problem, the tasks that can not be simply decomposed into a list of subtasks such as lifting a part of a large block. It can, however, be lifted by two robots from two sides. Therefore the task can be completed with two robots executing a plan of lifting together from both sides of the block. The CPS framework has a planning process where such an executable plan can be generated. The second problem of manager dependence is solved by inputting tasks to all the agents directly and not to a central manager. This is achievable because broadcasting does not need to specify an agent's eligibility. So all the agents in the system can select tasks according to their personal status and intention. In this way one agent's failure to perform will not affect the overall task's performance and the performance of the system. Addressing the last problem, regarding the coherent behaviour of a performer, the CPS framework explicitly states commitments that an agent has made to the team and the conventions under which an agent is expected to behave in an unexpected circumstances. When an agent fails to perform its bidded task, it will inform the task manager, and the task manager will re-advertise the task to the public. In this way the task will not be left

unperformed if there is an agent who is able to perform it. The system as a whole can achieve coherent behaviour.

4.4.2 Problems and Difficulties

The test results show the refinement of the CPS framework over the Contract Nets. However, there are problems perceived by the tests.

The first problem, somewhat unexpectedly, is the failure to perform an atomic task "bend_over" (see the results in Table 4.2). The problem occurred when agent2, which was configured as "prefers assistance" as its team formation condition, firstly received the task and it proposed a team formation for that task because it prefers assistance. When the task appeared on the task_to_bid partition of the blackboard, agent3 picked up the task; it also proposed a team formation to perform this task since it also prefers assistance. The system went on and engaged in a loop whereby the task never came to be performed. Studying the original team formation conditions in the CPS framework, it may be assumed that there are other agents in the system that do not prefer assistance. Otherwise a question is raised here whether simply specifying "prefers assistance" is adequate for the team formation proposing. The test indicates that it is inadequate. Unfortunately the CPS framework does not provide any definitions as to which agents there should be, and under what circumstances they should prefer assistance.

The second problem perceived during the testing is the difficulty in writing agents' team formation rules. The CPS framework indicates team formation as "the agent that recognised the potential for cooperative action solicits assistance in a form of team action. If it is successful, then it will end with a group having a joint commitment to collective action." Difficulty was experienced in designing the agents' rules for team formation since the CPS framework does not provide any strategy that an agent can use

to make a decision as to whether or not to join a team. In order to ensure that a team formation is successful a compromise method was adopted in which the rules in the team registration KS were designed to respond whenever possible. This means that all the agents adopt a "benevolent" stance. This benevolent stance can make team formation successful but it raises a concern about the drawbacks of the "benevolent" stance, such as extra communication cost and lower efficiency of the systemic behaviour. In the application of controlling multiple robots' cooperation, the robots have different but overlapping functions. We do not want less efficient robots to become involved in a cooperative action instead of more efficient robots. At least we should not allow the less efficient robots to have more chance than the more efficient robots to be involved in a cooperative action if they are available. A further investigation in this aspect needs to be taken.

The third issue, regarding the conditions for a cooperative action in the CPS framework, is regarded as too "narrow" in the sense that an all-agreed execution plan may not be necessary. The CPS framework states that a collective action cannot begin until the group agrees on a common plan. During the tests, the system initially suffered from the extra delay from performing atomic and combined tasks since the team formation and plan formation circles were performed. Actually atomic tasks do not need planning at all. Later in the system more agents were set to "no planning" for performing atomic and combined tasks. This is not a single and test-specific stance. The scenario does bring out a common case of teamwork, which may have important applications. Such teamwork includes "designed or orchestrated cooperation[1]" among cognitive systems. In these cases an overall plan, which all the members have agreed to, is not necessary.

[1] These terms are quoted from [Castelfranchi and Conte 96] to mean the cooperative systems that have no overall plan agreed by all team members before the cooperative action can be carried out.

The members of a team may not even be aware of the overall plan (for example, the members of an army command during an attack); nor are they always informed about the overall plan (think of partisans, guerrillas, and terrorists). Their actions form part of the same larger plan. They are coordinated with one another, necessary to one another, and act deliberately. In the implementation and tests, different planning schemes were used to provide flexibility. This is beyond the specification of the CPS framework.

The last issue regarding the criterion for successful team formation in the CPS framework, which is identified as the establishment of joint commitment among the team members, is too broad. The concern is raised by the heuristics perceived during the tests. When performing a complex task, team members only share the social rules that specify the commitments and conventions. There are other attributes that are important for a team in order to improve efficiency of the team. Such as resources sharing and a common measure of performance. The former is regarded as one of high intelligent team behaviours and the latter is regarded as necessary for measuring the performance of team members in case assistance and other rescue methods can be applied. Thus using only joint commitment as the only means of identifying team formation may not be enough.

4.4.3 Heuristics

The heuristics were generated during the implementation and testing. One of the common objectives of the CPS framework and this research is to try to develop a general model of multi-agent cooperation. It should be a full description of the multi-agent cooperation process. In describing multi-agent cooperation, the CPS framework offers a good attempt but not a complete description. The difficulties experienced in the implementation and the tests of the CPS framework revealed that other important processes involved in cooperative action have been neglected by the CPS framework.

The CPS framework does not address an agent's goal selection, in terms of describing why an agent has a goal and why an agent adopts other agents' goals by offering assistance to other agents. It does not provide any mechanism for goal selection. In the test a goal is a task that an agent chooses to perform. Neglecting agents' goal selection brought difficulties in the implementation in terms of programming agents' strategy for selecting a task, responding to a team formation proposal, proposing a task performance plan and offering a bid. The CPS framework also does not address act selection by agents specifying why an agent chooses to act in teamwork and why an agent chooses a particular teamwork if there are alternatives available. Neglecting act selection brought problems with the agents' act selection flexibility and behaviour rationality. In the test, if an agent chooses to perform a task in teamwork, it will remain with this team effort no matter how much the team formation will cost. Actually, in many cases teamwork is not the best way to accomplish an intended task. The performance of a team is affected by many factors such as the intention and the performance of the team members, the environment, the team structure and the policy adopted in the team [Steers 77]. If these factors combine together against the initial expectation of the team proposer, it is rational that the team proposer should drop its team formation efforts and try other alternatives. The problems perceived in the implementation and tests suggest that a complete description of cooperative action should include *goal selection* and *act selection*. With them various aspects associated with an agent's rational behaviour which result in a systemic efficiency can be characterised. Thus agents can choose and behave consciously both in self and social context.

4.5 Summary

Implementing the CPS framework intended to solve problems perceived in the implementation of the Contract Net framework and to develop a complete description

about a cooperation process among multiple autonomous agents. Studying the main differences and the improvements of the CPS framework over the Contract Nets, which are its separation of cooperation into a four-stage model rather than a single interaction, allows many aspects of the cooperation process to be studied and evaluated.

In the control of cooperation between multiple robots, the implementation of the CPS framework resolves the problems perceived in the implementation of the Contract Nets. Its applicability to the control of cooperative robots was obtained by using a number of compromised methods. The implementation revealed a number of aspects to which the CPS framework does not provide satisfactory answers. Firstly, the conditions of an agent's recognition for a potential cooperative action are not appropriate. Secondly, the CPS framework assumes that the agents in the system are benevolent. This agents' benevolent assumption prevents the full potential of autonomous agents from being exploited. Considering the CPS framework as a general multi-agent system model, besides its shortcomings associated with the details in each stage of the problem solving, it neglects other processing stages in cooperative problem solving such as goal selection and act selection. These deficiencies were highlighted by the difficulties experienced in the implementation of the CPS framework.

The last two chapters presented the initial experience of developing a framework for multi-agent cooperation. The method adopted in the development has been identified as *learning by doing*. The learning is achieved through the actual building of multi-agent systems, implementing existing frameworks and testing their applicability in the problem domain. The lessons are identified problems, difficulties and heuristics associated with the adopted frameworks, which were perceived during the learning process:

1. The Contract Net framework is inapplicable to the control of multiple robots' cooperation, which means it is unsuitable as a general model for multi-agent cooperation. It does not specify any social laws in a social context. Nevertheless, it provides a powerful communication protocol between agents that can be used among a group of agents for any social behaviour.
2. The CPS framework is applicable to the control of multiple robot cooperation with some compromised methods in place. It is a useful attempt to describe a multi-agent cooperation system that eases the difficulties of formalising a multi-agent cooperation model. For example, it identifies the conditions under which there is potential for cooperative action. As a general multi-agent system model, the CPS framework is not yet complete. It neglects other stages in a cooperative action that are important for system performance in a social context.

The implementation of existing cooperation frameworks not only provided useful experience in building multi-agent system but also provided a fresh way of developing a framework for multi-agent system. This is that both the Contract Net framework and the CPS framework were developed given inspiration by natural multi-agent systems. For example, the Contract Nets simulate human experts working together trying to complete a large task and the CPS framework developer found that human societies are natural multi-agent systems. Indeed, cooperative problem solving is a common process in our every day life, such as moving a heavy object, playing a symphony, or writing a joint paper. The next chapter will present a framework for multi-agent cooperation. It is greatly inspired by sociology. This method has been identified as *learning by analogy* in the beginning of this chapter.

Chapter 5

Shifting Matrix Management

Learning by analogy --

A good army commander has to learn to become a good Chess player.

-- A Chinese proverb

Human organisations are arrangements of distributed real intelligence. So, any model of Distributed Artificial Intelligence (DAI) is in some sense a model of an organisation.

-- Michael Masuch, 1992.

Starting from this chapter, the following three chapters will provide the main theoretical contributions of this thesis. Firstly, a Shifting Matrix Management (SMM) framework is proposed in this chapter. It is driven by searching for solutions to the problems perceived in the previous implementations of the existing cooperation frameworks. The SMM framework proposal is based on a fundamental belief that *any multi-agent system is a form of organisation*, and such an organisation is situated in an environment that can be specified for the purposes of analysis and evaluation. Thus a method of *learning by analogy* has been adopted. The name of the SMM framework is borrowed from

Mintzberg's theory of organisational structures [Mintzberg 79]. It describes a natural cooperation process among multiple autonomous agents in modern marketing enterprises. There is increasing evidence to support the claim that the framework of natural multi-agent systems in cooperative problem solving has a great impact on the artificial multi-agent systems [Wooldridge and Jennings 94(2), Castelfranchi and Conte 96]. The SMM framework provides a more complete description of a cooperative process among a group of autonomous agents and addresses a number of basic issues that are associated with the process of cooperation.

Chapter 6, as the second part of the theoretical contributions of this thesis, will provide a decision theory underlying an agent's act selection. The theory is developed because it has hitherto been neglected by DAI theorists. Without it, a complete description of the cooperation process can not be produced.

Finally, in Chapter 7 a formalisation of the SMM framework is provided based on the newly developed decision theory and the existing multi-modal logic. The formalisation of the SMM framework provides a computationally tractable multi-agent system model and offers a clear mapping from the theory to its implementation.

In this chapter, section 5.1 provides a brief review of organisation theory. It serves as an analogy that can be used to develop a framework for multi-agent systems since human beings can be viewed as intelligent autonomous agents and human organisations as natural multi-agent systems. In section 5.2 organisational approaches in DAI are reviewed. It studies existing efforts inspired by organisation theory in solving problems in multi-agent systems. The review of organisational approaches and organisation theory together provides the basis for the SMM framework as described in section 5.3. Finally section 5.4 provides a summary of the chapter and concludes with the need for a

decision theory about an agent's act selection in order for a formal SMM model to be developed.

5.1 Organisation Theory

Organisation theory studies "patterns of activities through which the task of the organisation is performed" [Miller and Rice 67, P. 33]. It has a long history of research in which many different patterns have been constructed [Miller 59, Trist et al. 63, Hall 72]. In organisation theory a pattern means "a set of administrative arrangements to cope with a given task" [Rice 65]. The initial motivation for developing different patterns of organisation was to find the most efficient arrangements for performing the task of organisation [Steers 77].

5.1.1 What is an Organisation

1. An organisation as an open system

One of the definitions of an organisation treats an organisation as an open system. An open system is "a set of elements standing in interrelation among themselves and with the environment" [von Bertalanffy 72, P. 417]. The general notion of an open system is simple and is composed of three components: inputs, throughputs or conversion, and outputs (see Figure 5.1).

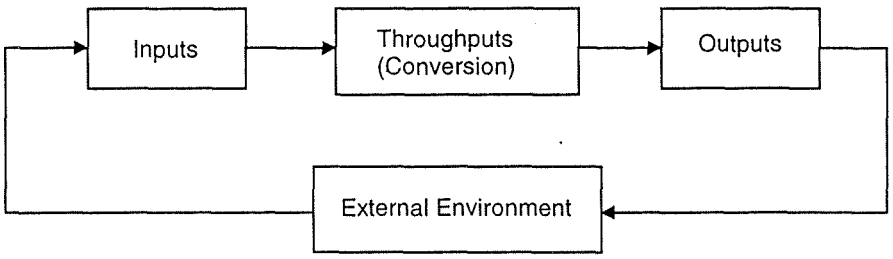


Figure 5.1 A basic paradigm for an open system model

Inputs represent all the factors that are "invested" in an organisation by the external environment. Such inputs may include new employees, money, raw materials, new machines, and so forth. These inputs are then transformed in the throughput or conversion stage into a variety of outputs that are returned to the environment, such as finished products, profits or return on investment, retiring or terminating employees, and so forth. Thus, an organisation is seen as continually interacting with its environment in a variety of exchange relationships.

Viewing an organisation as an open system can help us to understand the relationship between an organisation and its environment. This is that any organisation, as an open system, must exchange materials with its environment in order to survive [Rice 76]. To survive in a changing environment means maintaining its existence which further means keeping its three components. The exchange of materials from an organisation with its environment is through the process of *import-conversion-export*. This process in an organisation has revealed the following main points:

1. The difference between what it imports and what it exports is a measure of the conversion activities of the organisation. A manufacturing company, for example, imports raw materials, converts them and exports finished products (and waste). For the outputs it receives a pay-off, from which it acquires more inputs. The inputs to a university, as another example, are students; and the outputs are graduates (and failures).

2. The inputs and outputs are the results of *import-conversion-export* processes that differentiate organisations from each other. Every organisation, however, may have many *import-conversion-export* processes. A manufacturing company, for example, recruits employees, assigns them to jobs, and sooner or later exports them through resignation, retirement, dismissal or death. It imports and

consumes power and stores; it collects data about markets, competitors and suppliers' performance and converts the data into plans, designs and decisions about products and prices.

3. The nature of the many processes and their inputs and outputs reveal the variety of relationships that an organisation, or part of it, makes, with different parts of its environment, within itself, and between its different parts.

4. The different processes reveal the variety of tasks the organisation performs as a whole and the contributions of its different parts to the whole.

Based on the open system framework, organisation theory concludes that every organisation has, however, at any given time a *primary task* - the task it must perform to survive[1]. The *import-conversion-export* process is that process by which the primary task is performed. It is this dominant process that defines the essential relationship of an organisation to its environment, and to which other tasks and other throughputs are subordinate.

2. An organisation as a goal-seeking system

Definitions of organisations abound in the literature of organisation theory. For instance, Barnard [Barnard 38] views an organisation as "a system of consciously coordinated activities of two or more persons." In other words, organisations have stated purposes, communication systems and other coordinating processes, and a group of people who are willing to cooperate on the tasks necessary for goal attainment. Similarly, Etzioni [Etzioni 64] suggests that organisation is "planned units, deliberately structured for the purpose of attaining specific goals." Moreover, Porter, Lawler, and Hackman [Porter et al. 75] suggested that organisations are generally characterised by five basic factors: 1)

[1] For a fuller description of this concept and the system theory of organisation see [Miller and Rice 67].

social composition; 2) goal orientation; 3) differentiated functions; 4) intended rational coordination; and 5) continuity through time.

Several common threads run through these various definitions of an organisation. Most importantly, a common view of an organisation is the related notion of *goal orientation.* In other words, individuals are viewed as joining together and coordinating their activities to create a viable system capable of attaining common objectives. Every member may not value all the objectives equally; instead, an individual would probably pursue some goals which have less value to themselves in *exchange* for securing the efforts of others on those goals that are more highly valued by the individual. Thus, through coalition and cooperation, members of an organisation are attempting to satisfy their diverse needs and goals to the extent possible commensurate with resources.

Viewing an organisation as a goal-seeking system has several advantages. To begin with, it focuses attention on the dynamic interactions among the factors that affect the organisation's behaviours. For instance, we may examine how variations in tasks affect the organisational structure, and how these two variables (the task and the structure) jointly affect performance of the organisation. Secondly, by focusing on goals and objectives, it becomes possible to examine effectiveness and efficiency against organisational intentions. Instead of using our own value judgements concerning what an organisation should be pursuing, we can study within a more objective framework what an organisation is actually trying to do, and how well it succeeds [Steers 77]. Finally, viewing an organisation as a goal-seeking system allows for a clear recognition of the transient and changing nature of the organisation and its purposes. Not only does the organisation become modified over time in terms of structure but also in addition the goals that the organisation pursues may shift for various reasons. Hence, viewing an organisation as a goal-seeking system allows for a more thorough examination of how

an organisation is constructed and how its objective can be generated and pursued in a successful or unsuccessful fashion.

5.1.2 Organisation Existence and Performance

In the above open system view and goal-seeking view of an organisation, goal attainment is important. This goal includes performing the *primary task*. In order to survive and maintain some degree of effectiveness in an organisation, it is necessary to identify and study the factors that affect an organisation's existence and performance. Organisation theory suggests many factors that have a direct impact on the organisation's existence and performances. The most influential four factors are *organisational characteristics*, *environment characteristics*, *individual characteristics*, and *coordinating policies* [Steers 77].

- *Organisational characteristics*. It is the most important factor that affects the existence and performance of an organisation. Organisational characteristics mainly reflect the *structure* of an organisation. The *structure* refers to the relatively fixed relationship that exists in an organisation with respect to the arrangement of human resources. It is the unique way an organisation fits its people together to create an organisation. As such, the notion of structure includes such factors as the extent of decentralised control, the amount of task specialisation, the extent to which interpersonal interactions are formalised, and so forth. Thus structure defines how people are grouped together for the accomplishment of tasks.

- *Environment characteristics*. The second factor that affects the existence and performance of an organisation is the environment in which the organisation finds itself. The environment is the external forces that arise outside the

organisational boundaries and affect internal organisational decisions and behaviours. It includes the relative degree of environmental stability, the degree of environmental complexity, and the degree of environmental uncertainty.

- *Individual characteristics.* The third factor is the individual employee characteristics. This refers to the role of individual differences across employees in an organisation. Different employees possess different outlooks, goals, desires, beliefs, and abilities. These individual variations often cause people to behave differently from one another. These differences can have a direct bearing on two important organisational processes that can further have a marked impact on organisational effectiveness. These are *organisational attachment*, or the extent to which employees identify with their employer in different organisations, and in different individual *job performance*.

- *Coordinating policies.* Finally, coordinating policies employed and practised in an organisation, play a central role in the success of an organisation coordinating and facilitating goal-directed activities. These policies affect individuals' motivation, goals and behaviour. A suitable reward policy can attract more potential employees and motivate employees to contribute more to the organisation so that employees can satisfy personal needs and goals while simultaneously pursuing organisational objectives.

5.1.3 Diversity in Organisation

Organisations, sitting in different environments, performing different tasks, practising different coordinating policies, employing different people, are heterogeneous in size and shape. The diversity of organisations is identical with their structures. Here the *structure* refers to the manner in which an organisation organises its human resources

for goal-directed activities. It is the way the human parts of an organisation, which constitute the performers of activities, are fitted into relatively fixed relationships that largely define patterns of interaction, coordination, and task-oriented behaviour.

1. Linear and hierarchical structures

The earliest organisation structures result in a natural grouping in order to survive or to complete a particular task. It can be viewed as flat or linear structure [Steers 77, Mintzberg 79]. Later the hierarchy of authority and the distribution of power form a hierarchical structure. It starts with individuals being grouped into units, each under its own authority; then units are clustered into ever larger units under their own authority; repeating this clustering process until the whole organisation comes under a single authority. This hierarchical structure (see Figure 5.2) is still widely used in present organisations.

However, grouping is not simply a convenience for the sake of creating an organisation, or a useful way to keep track of everyone who works for the organisation. Rather, it is a fundamental way to coordinate works in the organisation for four reasons [Mintzberg 79]:

1. It establishes a system of common supervision among individuals and units.
2. It creates a common base of sharing resources among individuals and units.
3. It establishes a common measure of performance among individuals and units.
4. It encourages mutual adjustment among individuals and units.

Based on these regards, hierarchical structures always face the problem of which basis for grouping is to be used at the next level in the hierarchy. Individuals and units can be grouped by *function* which include knowledge, skill, work experience, and work

process, and by *market* which include output, client, and place. Generally, on one hand, individuals and units can be grouped by *means*, by the intermediate functions the organisation uses to produce its final outputs, on the other hand, they can be grouped by *ends*, by the features of the markets served by the organisation such as the products or services it markets, the clients it serves, the places where it serves them. No matter what the basis of grouping at one level in a hierarchical organisation, some interdependencies always remain. Functional groupings pose workflow problems, whereas market-based groupings impede contacts among like specialists.

There are three ways to deal with the "residual interdependencies":

1. a different basis of grouping can be used at the next level in the hierarchy;
2. a parallel grouping can be used. For example, staff units can be formed next to line units to advise on the problem;
3. a special device called *liaison,* coordinating the work of two units directly without having to pass through other control or managerial channels, can be overlaid on the grouping.

They are shown in Figure 5.2. In each case, however, one basis of grouping is always favoured over the others. The problems therefore still remain.

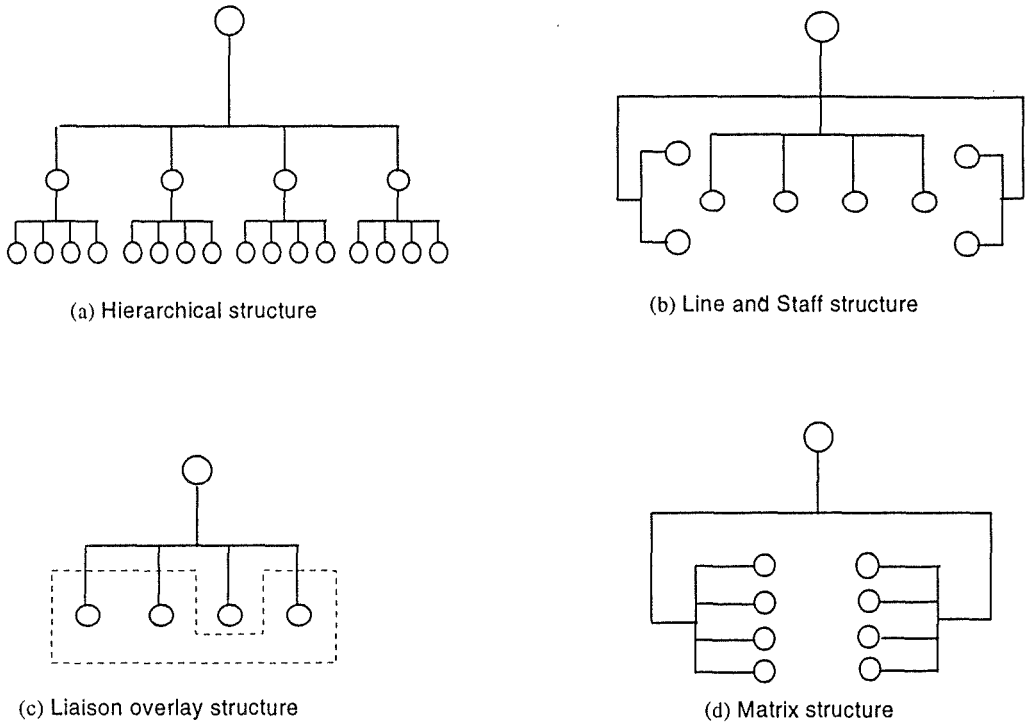


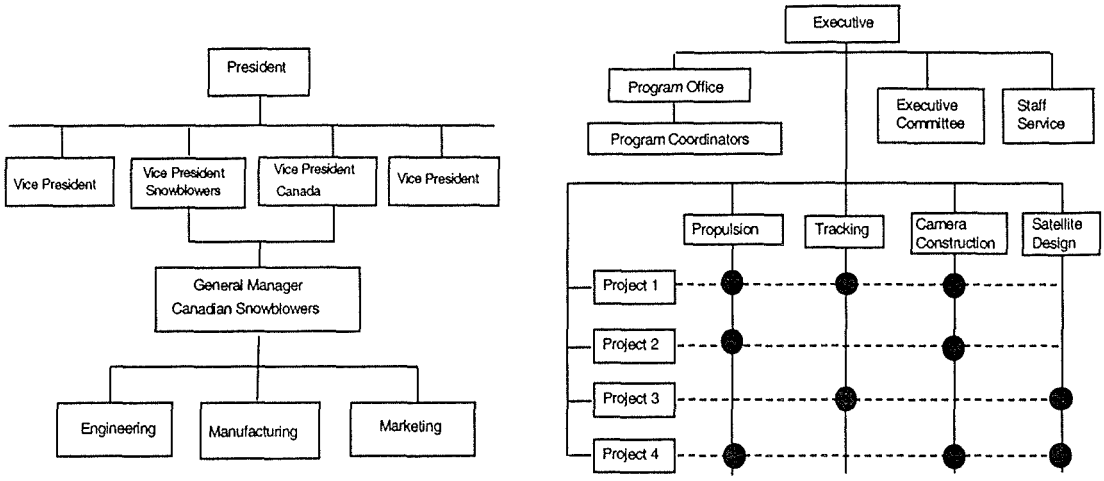
Figure 5.2 Structures to deal with residual interdependencies [Mintzberg 79]

2. Matrix and shifting matrix structure

The *matrix structure* (see Figure 5.2 (d)) is a modern structure found in many high technology organisations. It has been shown that it has many advantages over other structures such as linear and hierarchical structures [Mintzberg 79]. For example, dealing with the "residual interdependencies" problem, a matrix structure can compromise two or more bases for grouping together rather than a single one. In general, the matrix structure is a structure that is used to balance two or more bases of grouping, as shown in Figure 5.2(d), for example functional with market or one kind of function with another, or one kind of market with another kind of market. This is done by the creation of two or more dimensions of authority, constraint, and responsibility in an organisation.

A *shifting matrix structure* is defined in contrast to a *permanent* form of matrix structure in which the individuals remain more or less in a fixed place as shown in the example of

a hypothetical multinational firm in Figure 5.3(a).



(a) A Permanent Matrix Strucutre in An International Firm

(b) Shiftting matrix structure in the NASA weather satellite program

Figure 4.3 Matrix and shifting matrix structures [Mintzberg 79]

In Figure 5.3(a) the position of an individual in the organisation is relatively fixed. In a shifting matrix structure the units and the individuals move around frequently depending on the tasks of the organisation in hand. A typical example of the shifting matrix structure in a high technology industry can be constructed by a combination of function and the task. That is individuals are grouped by speciality in functional departments for housekeeping purposes and deployed them from various departments in task performance teams to do the work, as shown in Figure 5.3 (b).

5.1.4 Summary

Organisation theory studies the patterns of activities through which the tasks of the organisation are performed. The two dominant views of an organisation are the *open system view* and the *goal-seeking system view*. The open system view emphasises the response of an organisation to its environment. It concludes that any arrangement of organisational components and their relationship has to be in favour of accomplishment of the *primary task*. The primary task is the task that organisation must perform to survive in the environment. It is the performance of the primary task that defines the essential relationship of an organisation to its environment, and to which other tasks and

other performances are subordinate. The goal-seeking system view emphasises the ultimate goal of the organisation, which is the attainment of the common objectives of the organisation. A goal-seeking system view enables a study of various parts of an organisation and the organisation itself can be taken. The two views of organisation are not contradictory but complementary. On one hand, to attain the common goal of an organisation, the organisation has to survive in the environment first. On the other hand, the purpose of survival of an organisation is to perform activities and to attain its intended goals.

Based on the two views of an organisation, the four factors that have the greatest impact on organisational existence and effectiveness have been identified. These are structure, environment, individuals and policies applied in the organisation. Among these four main factors, structure is the most important factor that affects the others. Four types of structure linear, hierarchy, matrix and shifting matrix have been reviewed.

Returning to the initial purpose of reviewing organisation theory, as a means of finding a solution for multi-agent cooperation problems. Organisation theory, as described above, shows how the different views of an organisation; the factors that affect the existence and the effectiveness of an organisation; the existing structures in different organisations, can be beneficial in searching for a solution for the cooperation problems in multi-agent systems. Above all, the most important lesson that should be learnt from organisation theory is that the pattern of arranging activities in an organisation for its goal attainment is adopted by the basic entity of the organisation rather than an external force. Therefore, research efforts in multi-agent systems should not be focused on designing a fixed structure or system arrangement for a specific multi-agent system. Instead the efforts should be focused on the study of the mechanisms that allow agents to organise a structure or a pattern by themselves. If one does so, one can confidently

believe that the agents in any multi-agent systems should have a similar structure or pattern with those in human organisations since we treat agents as autonomous and intelligent entities.

5.2 Organisational Approaches in DAI

Learning from organisation theory is not a new idea. Many researchers in DAI have already done so. This section provides a review of existing work related to learning from organisation theory in DAI. In the first part, the definition of organisation and the definition of structure in DAI are introduced. The second part of this section outlines the significant research in organisational study in DAI. Finally, the problems existing in the current approaches are identified in order for the proposal of the SMM framework to be placed in context.

5.2.1 Organisation and Structure Definitions in DAI

Historically, DAI systems have had very limited flexibility to adapt their global behaviour, since usually they were designed to handle a single aspect of the problem at hand, such as task allocation [Cammarata 83, Davis and smith 83, Durfee and Lesser 87]. In such systems, there is no notion of an organisation. At the most it only has a notion of structure. This structure is a fixed arrangement of different parts in a system. Until, later, the problems that DAI faced reached such a level that complexity, distribution, dynamic, and openness could not be ignored, then it became important to develop DAI systems to be flexible, adaptable and incremental. The multi-agent systems approach is one of the efforts to build such systems. It adopts a bottom-up design strategy and much effort has been given to studying individual agent's attributes. Nevertheless, the notion of organisation has been introduced. Unfortunately the dominant philosophy in multi-agent systems research is that *the success of the*

individual agent is the sole metric for the evaluation of the system performance [Moulin and Chaib-Draa 96]. It is anything but a complete view according to organisation theory.

Based on this incomplete view, organisation and structure have different definitions to the ones in organisation theory. Some definitions of organisation in DAI are given below:

- Gasser [Gasser 86] views an organisation as a "particular set of settled and unsettled questions about beliefs and actions through which agents view other agents."
- Bond defines organisation as a set of agents with consistent mutual commitments [Bond 90].
- Malone's [Malone 90] definition is "a group of agents is an organisation if they are connected in some way (arranged systematically) and their combined activities result in something better (more harmonious) than if they were not connected."
- Moulin and Chaib-Draa's [Moulin and Chaib-Draa 96] explanation is "a set of agents with mutual commitments, global commitments, mutual beliefs, and eventually, joint intentions when these agents act together to achieve a given goal."

The difference between the definitions in DAI and in organisation theory is that in DAI the definition of organisation is based on notions such as beliefs, intentions, and commitments. They are driven by the degree of cooperation that exists between agents, and by the spectrum of communication strategies that is offered to an agent in order to exchange beliefs, intentions, and commitments [Moulin and Chaib-Draa 96].

A *structure* in DAI has been defined as a pattern of information and control relationships that exist between agents, and the distribution of problem-solving capabilities among them [Moulin and Chaib-draa 96]. In cooperative, distributed problem solving for example, a structure gives each agent a high-level view of how the group solves problems and the roles that each of the agents plays within the structure. With this view, the agents can ensure that they meet conditions that are essential to successful problem solving, including the following [Corkill and Lesser 83]:

- *Coverage*: all necessary portions of the overall problem must be included in the activities of at least one agent.
- *Connectivity*: agents must interact in a manner, which permits the covering activities to be developed and integrated into an overall solution.
- *Capability*: coverage and connectivity must be achievable within the network's communication and computation resource limitations, as well as the reliability specifications of the group.

Generally, the structure must specify *roles* and *relationships* to meet these conditions. For example, to ensure coverage, the structure could assign roles to agents according to their competence and knowledge of a given subproblem. The structure must also indicate the information of connectivity to the agents so that they can distribute subproblems to competent agents. This connectivity information should also allow agents with overlapping competence to avoid duplication of effort when solving the same subproblem.

It is clear that in DAI, organisation arises as a result of joint actions from agents' mental behaviour and the environment. Structure is a pattern defined by external sources as a high-level constraint to agents to ensure the success of problem solving.

5.2.2 Societies of Agents in DAI

Many researchers in DAI realise that: 1) human organisations are naturally distributed systems; 2) there have developed various kinds of solutions for the problems faced by the individuals who belong to them, such as distributed decision making, hierarchical control, group coordination activities, etc.; 3) their results can be usefully adopted by DAI research [March 65, Bond and Gasser 92, Moulin and Chaib-draa 96]. They proposed and developed several organisational models for multi-agent systems or models which are focused on a part of organisation activities or goal attainment process that can be used in multi-agent systems [March 65, Fox 81].

1. Research related to organisation characteristics

Kornfeld and Hewitt - *Scientific community.* Kornfeld and Hewitt have proposed that multi-agent systems can be organised in a manner analogous to the organisation of scientific research [Kornfeld and Hewitt 81]. In their *scientific community metaphor*, agents are organised into loose classes that have general problem-solving goals. *Proposers* propose possible solutions to the problem at hand. *Proponents* then collect and present evidence in favour of a proposal, while *sceptics* collect and present evidence to disprove it. *Sponsors*, or *evaluators*, examine proposals as they accrue and balance the system so that more work is done on those proposals that seems to be favourable.

Malone - *Group organisation.* Malone [Malone 90] did a comprehensive study of group organisation. He analysed in a systemic way the analogies between human organisation and computer systems. He proposed a framework for analysing different ways of segmenting and coordinating tasks among a group of agents. He claimed that the group organisation depends upon the capacity of agents to coordinate their activities.

Wooldridge - A model of multi-agent systems. Wooldridge [Wooldridge 92, Wooldridge and Fisher 92] based on both natural and artificial multi-agent systems developed a family of logic for representing the properties of multi-agent systems. He also developed a simple, and in some sense general, model of multi-agent systems. His model has been implemented in the previous chapter as a part of the *learning by doing* approach in this research.

2. Research related to coordinating policies

Smith and Davis - *Contract-net.* Smith and Davis [Smith and Davis 81] developed a human organisation metaphor called Contract Nets for cooperative task performance. In the Contract Net framework, agents coordinate their activities through *contracts* to accomplish specific tasks. This metaphor has also been implemented in the previous chapter.

Bond - *Consistent mutual commitments.* Bond [Bond 90] uses the concept of commitment from sociology introduced by Becker [Becker 60]. Because agents' cooperative behaviour is regarded as that the agents participate in several organisations, the notion of commitments must be introduced since the agents may participate in other organisations and it is the notion of commitments that constrains the agents' cooperation. Bond also attempted to formulate this idea of commitment, and to extend it into the notions of agent and organisation.

Shoham and Tennenholtz - *Useful laws.* Shoham and Tennenholtz [Shoham and Tennenholtz 92] proposed a formal approach to deal with "useful laws for artificial agents' societies." These social perspectives of DAI greatly influenced the solutions proposed for modelling group activities such as cooperation or coordination of agents.

3. Research related to individual characteristics

Cohen and Levesque - *Persistent goal and intention.* Cohen and Levesque's contributions [Cohen and Levesque 90(1)] are not directly aimed at the agents' organisation since their formalism was originally used to develop a theory of intention which the authors required as a pre-requisite for a theory of *speech acts* [Austin 62, Searle 69, Cohen and Levesque 90(2)]. However, their formalism has subsequently proved to be useful for reasoning about agents. It has been used in an analysis of conflict and cooperation in multi-agent dialogue [Gallers 89], as well as in several studies in the theoretical foundations of cooperative problem solving [Levesque 90, Jennings 92, Castelfranchi 90, Rao and Georgeff 91, Singh 94, Wooldridge and Jennings 94]. Cohen and Levesque based on the philosophical work of Bratman [Bratman 90], constructed a *logic of rational agency*. Over this framework, they introduced a number of derived *constructs*, which constitute a "partial theory of rational action" for individual agents; *intention* is one of these *constructs*. The first major derived *construct* is the *persistent goal*. An agent has a persistent goal φ if and only if:

1. It has a goal that φ eventually becomes true, and believes that φ is not currently true.

2. Before it drops the goal φ, one of the following conditions must hold: (i) the agent believes φ has been satisfied; or (ii) the agent believes φ will never be satisfied.

It is a small step from a persistent goal to a first definition of intention, as in "intending to act": an agent intends to do action α if and only if it has a persistent goal to have brought about a state where in it believed it was about to do α, and then did α.

Rao and Georgeff - *joint intention.* Similarly, motivated by trying to describe, analyse, and specify individual agents' behaviour by *intentional* notions [Dennett 87], but different from Cohen and Levesgue's work, Rao and Georgeff [Rao and Georgeff 91,

93] are focused on describing individual agents collaborative working within a social context. The notion of *joint intention* was introduced to explain how team members could act together. In Rao and Georgeff's logical framework, three primitive modalities: *beliefs*, *desires*, and *intentions* were used. This is different from Cohen and Levesque's work where only two basic attitudes, *beliefs* and *goals*, were used, and intention was defined in terms of *beliefs* and *goals*.

Jennings and Mamdani - *joint responsibility.* Jennings and Mamdani [Jennings and Mamdani 92] view *joint responsibility* as a "metalevel" description of how cooperating agents should behave when engaged in collaborative problem solving. Their formal account of *joint responsibility* uses modal and temporal logic to define preconditions. These preconditions must be satisfied before joint problem solving can start. They also prescribe how individual team members should behave once joint problem solving has started: agents agree that they will obey a "code of conduct" to guide their actions and interactions while performing the joint activity.

4. Other research influenced by organisation theory

Werner - *Unified theory for social unit.* Werner [Werner 89] developed a unified theory of communication, cooperation, and social structure. It is used as the foundation for the design of agents' systems that behave as a social unit or group. He proposed a formal account of an agent's intentional states and related the linguistic messages to their effects on the planning process. This approach allowed him to give an account of social cooperative action, because agents' intentional states are mutually modified by communicative exchange, for example a conversation or a discourse.

Star - *Structure of ill-structured solution.* Star [Star 89] supported the idea that the development of DAI should be based on a social metaphor, rather than on a

psychological one, and suggested that systems should be tested regarding their ability to meet community goals. Star suggested to use the concept of "boundary objects" derived from analysis of organisational problem solving in scientific communities. Boundary objects are those objects that are malleable enough to be adaptable across multiple viewpoints, yet maintain continuity of identity.

Gasser et al. - *Settled and unsettled questions.* Gasser and his colleagues [Gasser et al., 89] developed a framework for representing and using organisational knowledge in DAI. They view organisation as a coordination mechanism, a particular set of settled and unsettled questions about belief and action that agents have about other agents. The way to define organisation is to locate the concept of organisation in the beliefs, expectations, and commitments of agents themselves. With this viewpoint, organisational change means opening and/or settling some different sets of questions in a different way, giving individual agents new problems to solve and, more importantly, different assumptions about the beliefs and actions of other agents.

There are other comparisons with various human organisations that provide a qualitative insight for DAI. These include *economic markets* [Miller and Drexler 1988], *the Society of Mind* [Minsky 86], and *social dependence* [Castelfranchi 92].

5.2.3 Summary

DAI initially does not have the notion of organisation. This is because of the nature of the problems that it tries to solve and the way it solves the problem. Usually the problems, although they are mostly distributed, are also clearly defined. The system thus has a clearly defined boundary, function and capacity. The dynamics, openness and adaptability were not problems in designing such DAI systems.

Until the problem became so complex that the boundary of the problem-solving system cannot be clearly defined before the system can be constructed, multi-agent systems was introduced and the notion of agents' organisation was also introduced. However, the definition of agents' organisation was different from the definition of organisation in the organisation theory. This is because the multi-agent systems approach was focused on the individual agents and was trying to represent an agent in a more rational way so that the agent can perform rationally in a social context. The dominant philosophy in multi-agent systems was that the success of a multi-agent system performance depends on the success of an individual agent's performance. It ignored other factors such as an agent's organisational characteristics. Based on this incomplete view, a structure in an organisation became a top-level constraint on an individual agent. Agents only played roles that were defined by the structure within a predefined relationship.

Efforts in studying human society, particularly those having an inspiration from organisation theory, enabled many multi-agent cooperation models to be proposed, such as *scientific community* by Kornfeld and Hewitt and a *model of multi-agent systems* by Wooldridge. However, several models were specially focused on one process in an organisation's goal attainment. For example, the *Contract Net* by Smith and Davis and the *group organisation* by Malone were models for task allocation. Others were focused on agents' coordinating policies and individual agents. They missed a fundamental point from organisation theory, which is that the structure of an organisation ought to be designed, chosen, selected and adopted by the basic entities of the organisation. In the human organisation the basic entities are human, so the organisational structure is studied and designed by humans themselves. Similarly, in multi-agent systems, the structure of agents' organisation should be designed, constructed, or adopted by agents. It does not mean that we cannot discuss and study agents' organisational structure.

Instead, this author believes if the agents are really autonomous and intelligent, they should be able to adopt the results from humans as high intelligent agents, or they should have a similar structure in their organisation as we do in human organisation. The point is that we should provide agents enough knowledge, logic, intelligence, and autonomy to enable them to structure their organisation in a similar format to humans, rather than copy human organisation structures and apply them to multi-agent systems.

Research in constructing and representing an intelligent agent has achieved many useful results, such as *intention* theory [Dennett 87, Cohen and Levesque 90(1), (2), Rao and Georgeff 91, 93] which is based on *folk psychology* and *anthropomorphism*. The former states that human behaviour is predicted and explained through the attribution of attitudes, such as believing, wanting, hoping, fearing and so forth. The latter offers a way of representing agents in terms of human-like mental states. Firstly, there are a number of researchers who have proved that an agent's intentional stance is appropriate [McCarthy 79, Shoham 93, Seel 89, Rosenschein and Kaelbling 86]. Secondly, the agents intentional stance explains how an individual agent's behaviour results from and is conducted by its mental states [Cohen and Levesque 90 (1) (2), Rao and Georgeff 91, 93]. Thirdly, in line with this research, the agent's intentional stance can help to explain a group of agents' organisational behaviours by their individual mental state. Therefore the SMM framework is not only inspired by organisation theory but also encouraged by similar works in multi-agent systems.

5.3 Shifting Matrix Management

In this section the SMM framework will be proposed based on the organisation theory and the related work in multi-agent systems. The available theories that could be possibly used to support the SMM framework are listed. The need of a decision theory

that governs an agent's act selection is identified which sets up an objective for the next chapter. The application of SMM in multi-agent systems is a novel contribution, first reported in [Li et al 97].

5.3.1 Shifting Matrix Management Framework

Shifting Matrix Management describes a natural cooperation scenario where:

> a *collective of agents*, with different *motives* and *capabilities*, gather together to perform *some tasks*. At any given time, the agents in the system always find that they are located in a cross point of a shifting matrix structure that is jointly defined by the tasks, the motive and the capability of the agents. Individual agents have the right to select tasks to work on according to their motive and capability by joining a task performance *team*. Working in isolation is viewed as a singular team. The team can also select its members by its existing members *suggesting* or *refusing* an individual to join the team. After a team is formed, a temporary, relative fixed relationship in the *team* is constructed. At the same time a commonly accepted *social norm*, *performance measurement, resources allocation* and *mutual adjustment* in the team are also established. Each member will act according to this social *norm* as a "code of conduct." A team can have different roles played by the individual members. Members have the right to contribute to the team in different forms such as propose, discuss, or determine a performance plan, or act as an executor in the task performance process and so forth. Once the task has been accomplished, the team is disbanded and the individual agents are free to organise or join other teams. Then a new task performance circle is started and individuals will experience the same process where they may find that they are located in different positions of the shifting matrix structure.

To describe this scenario, a six-stage framework is proposed for any multi-agent cooperation systems [Li et al. 97]. The six stages are:

1. **Goal selection**: The SMM model starts when a set of new tasks is generated in the multi-agent systems. There are a number of agents who distinguish themselves from other agents by different motives, functionality, and knowledge. These differences define the agents' variety of mental states in having goals, beliefs and intentions. The first action performed by agents in a cooperation process is to select *tasks to perform* related to their initial mental states. At the end of this stage, agents should have settled goals.

2. **Act Selection**: In this stage, agents will select and adopt a way in which their settled goals can be achieved. Normally, there is more than one way of achieving a goal. For example, an agent that recognises its intended goal is a common goal with other agents, would have to decide whether to bring about this goal in isolation or in teamwork with other agents. This decision making is the result of an agent's individual mental behaviour on alternative possible actions. It is affected by a number of factors, such as the chances of an action taking place and being completed, the personal evaluation on the consequences of the action, and the personal idea about the outcomes of the action. The termination of this stage is that the agents in the system have decided the way of achieving their intended goals.

3. **Team formation**: In this stage, the agents that decided to achieve their goals in a cooperative manner will attempt to organise the tasks' performance teams. If their attempts are successful the teams shall be formed. The establishment of a team means creation of a *common rule*, a *base of sharing resources* within the team, a *common measure of performance* and a *mutual adjustment* among team members.

4. **Plan formation**: One way of maximally utilising the resources and the functionality among team members is to have a *joint plan* that most members in the team believe is the most favourable way of achieving the intended goal. The workload among team members can then be re-distributed according to this *joint plan* so the best candidate will do the most suitable work. The plan formation is a process of negotiation so that the team members jointly attempt to reach a state where they agree on a settled way of achieving their joint intended goal.

5. **Team action**: In this stage, the newly agreed joint plan is executed by the team members. This execution is under the "code of conduct" that has been set up when the team is formed to ensure that the each member's share of the task in the joint plan can be successfully done.

6. **Shifting**: The last stage of a cooperation process is the agents' goal, position, and role shifting. The end of this stage marks the disbandment of the team. In this stage, *mutual adjustments* among agents take place despite the adjustments, which were carried out during other stages. This mutual adjustment emphasises learning from each other, knowing each other, and establishing 'trust', 'credit', or 'friendship' towards each other according to the success or the failure they experienced together. This is important because the agents' cooperation is not a one-go process. It is a circular process where each circle consists of the above six stages until all the tasks are accomplished. The agents *know about each other* will increase the chance of success in the next round of cooperation and thus improve the system performance.

5.3.2 The Theories Used in the SMM Framework

Only providing a general framework is not enough. Theoretical support is necessary to construct a formalised SMM model that can benefit both application and formal proofing. The purpose of this section is to seek the theoretical support for the SMM framework.

The six stages in the SMM framework can be viewed as three main layers: an individual layer which refers to the agents' *goal selection* stage, a pre-social layer which is composed of *act selection* and *team formation*, and a social layer which consists of *plan formation*, *team action* and *shifting*. The individual layer, as its name suggests, is merely a layer of individual agents' mental behaviour. The second layer is called the pre-social layer because although it is still at the stage of describing individual agent's mental and physical behaviour, its operands are no longer static, non-intelligent objects such as tasks. They are now agents. For example, in the stage of team formation, the agent, who recognises the potential and decides to achieve the goal in cooperation, will have to deal with its potential team mates in a form of proposing, persuading or convincing. However, it is not a 'real' social action layer as in the last layer. In the last layer, which is the social layer, after team formation, agents should behave in a full social manner.

The existing theories in multi-agent systems that deal with the problems in the above three layers and can be pinpointed and possibly adopted by the SMM framework are listed in Table 5.1.

Table 5.1 The theories could be adopted by the SMM framework

Layers	Stage	Theories	References
Individual	Goal selection	Intentional Models	*[Bratman 87, 90, Cohen and Levesque 90 (1), (2), (3)]*
Pre-social	Act selection	Intentional Models, Behaviour Models?	*[Bratman 87, 90, Cohen and Levesque 90 (1), (2), (3), Castelfranchi and Conte 96]*
	Team formation	Negotiation, Contract Net	*[Smith and Davis 81]*
Social	Planning	Negotiation, Contract Net	
	Team action	Joint Commitment and Social Convention	*[Jennings 92(2), 93]*
	Shifting	Learning	

In the individual layer, a modified version of Cohen and Levesque's *intentional models* could be adopted to explain the mental states of an individual agent's goal selection. In Cohen and Levesque's intentional model, if an agent intends to achieve an objective φ, then the following properties should hold [Bratman 87, Cohen and Levesque 90(1)(3)]:

- It believes that φ is possible.
- It does not believe that it will not bring about φ.
- Under certain conditions, it believes it will bring about φ.

In Cohen and Levesque's intentional model, only two attitudes are used. They are *beliefs* and *goals*. Further attitudes, such as intention and knowledge, are defined in terms of these two basic attitudes. The research presented in this thesis neither intends to join the arguments about the "true" definition of complicated cognitive and philosophical concepts such as motive, desire, goal, belief, intention, etc., nor intends to justify the relationship between them. Based on the practical experience of constructing multi-

agent systems, this research is only trying to provide a self-contained, practical and tractable theory of agents' cooperative behaviour by specifying that the agent's goal is directed by its initial *motive* and its *capability*. The agent's capability is further identified as *knowledge* and *ability*. The details of an agent's motive, belief, goal, intention and their formal account are provided in the SMM formalisation in Chapter 7.

In the *social layer*, agents shall behave in a social manner. This is that an agent, as a team member, can not only pursue personal goals but also needs to pursue organisational goals as well. Many researchers have noted that in a social setting both agent's mental states and behaviour should be governed by certain social laws [Cohen and Levesque 90(2), Rao et al. 92, Shoham and Tennenholtz 92, and Jennings 93]. Jennings [Jennings 93] based on the notion of joint intention given by Cohen and Levesque, proposed a model of joint responsibility. The theory specifies two critical notions of *commitment* and *convention* which together define how an agent should behave not only when things goes smoothly but also when things go wrong in a social setting. The model and the theory could be adopted to describe agent's behaviour in the social layer of the SMM framework. The details and the formal description are also provided in Chapter 7.

A problem occurs in finding a theory for the *pre-social* layer. A theory is needed to explain why agents shall choose to work in a cooperative manner against working in isolation or another cooperative way. The theory needs also to explain to what extent an agent shall abandon its efforts at organising a team by persuading other agents to work together, and to seek new partners instead. The fundamental question is *why are joint activities formed.* Answering this question, the two existing theories are *intentional theory* and *behaviour theory*. We shall examine these two theories in detail to show how they address the question and possible inadequacies in them.

Intentional theory [Cohen and Levesque 91, Rao et al. 92] proposes that *joint intentions* expressed by *joint persistent goals* is the basic factor of agents' joint activities.

In Cohen and Levesque's theory, the central notion of *joint intention* is defined through a notion of *joint persistent goals* which in turn are based upon the concept of *achievement goals.* Achievement goals define the state of individuals participating in a team which is working in a collective goal (e.g., moving a table[2]) with a specified motivation (e.g., to gain access to a cupboard). Agent i has a *weak achievement goal*, relative to its motivation ζ to bring about φ if either of the following are true:

- i does not yet believe that φ is true and has φ being eventually true as a goal (i.e., i has a normal achievement goal to bring about φ).
- i believes that φ is true, will never be true or is irrelevant (ζ is false), but has as a goal that the status of φ be mutually believed by all team members.

Thus a weak achievement goal involves four separate cases: either i has φ as a normal achievement goal (it wants the table to be moved); thinks that φ is true and wants to make this fact mutually believed (it believes the table has already been moved); believes that φ will never be true (it believes that the table is nailed to the floor) and wants to make this fact mutually believed or, finally, believes ζ is no longer true (there is no longer a need to gain access to the cupboard).

With these weak achievement goals, joint persistent goals are defined as a team of agents having a joint persistent goal, relative to ζ, to achieve φ if and only if:

1. they mutually believe that φ is currently false (e.g., the table has not been lifted);

[2] This example is quoted from [Jennings 92(2)].

2. they mutually believe that they all want φ to be eventually true (e.g., they all want the table to be lifted);

3. they will continue to mutually believe that they each have φ as a *weak achievement goal* relative to ζ until they come to mutually believe either that φ is true, that φ will never be true or that ζ is false.

The intentional theory "is not sufficient to account for a group or a truly cooperative work" [Castelfranchi and Conte 96]. An example given by Castelfranchi and Conte is used here to prove the point.

Consider Professor A in France and Professor B in the United States who share a goal φ (to discover an anti-AIDS vaccine) relative to a motivation ζ (eradication of AIDS). They share all three mental attitudes described above for a joint persistent goal and then for a team (cooperation):

1. They mutually believe that φ is currently false (a vaccine has not already been discovered).

2. They mutually know they all want φ to be eventually true (to discover the vaccine).

3. It is true (and mutually known) that until they come to believe either that φ is true, that φ will never be true, or that ζ is false, they will continue to mutually believe that they each have φ as a *weak achievement goal* relative to ζ and with respect to the team: they want that the status of φ (i.e., whether the vaccine was discovered or not) be mutually believed by all the team members, both because this is an expected practice within the scientific community, and because they want to inform their competitor.

No one would say that Professor A and Professor B form a team and work together. Indeed, they might come to strongly compete with each other. The reason for this is that the persistent goal in the example is shifted from a state of affairs (bring about a state of the world such as to discover the vaccine) to an actual action (cooperation). The theory then becomes clearly irrational because there are alternative actions. Cohen and Levesque also note that because an agent commits itself to a goal, it cannot be concluded that the agent will act in a particular way.

Behavioural theory, inspired by interdependence in human relationships in a social context, claims that *mutual dependence* defined by *social dependence* is the foundation of social interaction or cooperation. In this theory, agents cooperate if and only if they mutually believe that they are *mutually dependent* on each other to achieve a common goal [Jennings 92, Conte et al. 91]. The mutual dependence is defined by social dependence. Agent i_1 is dependent on agent i_2 to achieve goal φ with regards to an action α is defined in the following manner:

> *agent i_1 is dependent on agent i_2 with regards to an act useful in achieving goal φ when i_1 is unable to do the action α to realise φ while agent i_2 is able to do so.*

The behavioural theory stresses that "the social dependence is not fundamentally mental. It is an objective relationship, in that it holds *independently* of the agent's awareness of it." [Castelfranchi and Conte 96, page 537]. The mutual dependence is then defined by the social dependence. Agents i_1 and i_2 mutually depend on each other to achieve a common goal φ is defined as:

> *agents i_1 and i_2 depend on each other to achieve a common goal φ, which can be achieved by means of a plan including at least two different actions such*

that agent i_1 depends on agent i_2 doing action α_2 and agent i_2 depends on agent i_1 doing action α_1.

With this, Conte and his colleagues conclude: "cooperation is a function of mutual dependence: in cooperation agents depend on one another to achieve one and the same goal; they are co-interested in the convergent result of the common activity."

The problems with behavioural theory are associated with its lack of description of agents' mental states. The most salient is the *perception* or the awareness: although agents depend on each other to achieve a common goal, because of *unawareness* they can hardly be said to cooperate. Another difficulty associated with behavioural theory is the different structures of the dependence relationship in a social context. For example, Castelfranchi *et al.* revealed the existence of *OR-dependence*, a disjunctive composition of dependence relations and the *AND-dependence*, a conjunction of dependence relations. What consequence may be derived from agents (either unilaterally or mutually) becoming aware of it? Surely it will not be a single result of *working together* or *cooperation*. For example, agent i_1, who has a goal φ, perceives the dependence with agent i_2, with regards to agent i_2 doing action α, to achieve its goal φ. Agent i_1 may try to influence agent i_2 to pursue goal φ. Now agent i_2 may choose whether to adopt i_1's goal φ or otherwise. It is dependent on how agent i_2 accesses the dependence relations with regard to its own current goal. There are at least two cases that may result in agent i_2 not responding to cooperation with agent i_1: 1) agent i_2 is not aware of the mutual dependence, and 2) although agent i_2 is aware of the mutual dependence, however the goal that agent i_2 is dependent on agent i_1 to achieve is not the next important goal according to agent i_2's goal priority. The only chance that cooperation can be formed is that agent i_2 is aware of the mutual dependence and the goal that involving the mutual dependence is the next important goal to agent i_2.

From the above analysis, it is clear that both intentional theory and behavioural theory are inadequate to explain why agents joint activities are formed and to what extent agents should attempt actions that can bring about a joint activity. What is really needed in current multi-agent systems is a theory that can explain agent's act selection including joint activities.

Overall, in this section the SMM framework was proposed. It separates a multi-agent cooperation process into six stages. This separation enables other useful models that were developed in multi-agent systems to be studied and re-used as different stages of cooperation process. For example the Contract Net framework can be used for communication between agents. This separation also enables the needs of current theory in multi-agent systems to be identified. That is a theory which can explain why agents form a joint activity and to what extent an agent should abandon its effort in forming such an activity.

5.4 Summary

This chapter provides a different methodology for developing a framework for multi-agent cooperation, which is classified as *learning by analogy*. It is based on the previous approach of *learning by doing*. The *analogy* is organisation theory. It is chosen because the basic belief of this research is that humans are intelligent agents; human organisation is a natural multi-agent system; and the results of its research can be learnt for developing a new framework for multi-agent systems.

The first section of this chapter provided a review of organisation theory in which related results were provided. Two dominant views of an organisation were reviewed. Viewing an organisation as an open system enables a conclusion to be drawn so that the primary task is the dominant task to which other tasks are subordinate. It is the

performance of the primary task that defines the essential relationship of an organisation to its environment. Viewing an organisation as a goal-seeking system enables the relationship and the interaction among various parts of an organisation and between them and their environment to be studied. Based on these two views the four main factors that affect the existence and the goal attainment of an organisation were identified. The existing structures of organisations were also studied. The lessons learnt from organisation theory are:

1. The primary task to be performed in an organisation is to survive.
2. The relationships in an organisation should be in favour of its common goal attainment.
3. The main factors that affect an organisation's goal attainment are structure, environment, individual and policies.
4. Shifting matrix structure can balance more than one aspects of organisation's concern.

The most important point from organisation theory is that any organisational arrangement should be designed and constructed by its basic entities.

In the second section of this chapter, research that has the same source of inspiration as this research in DAI was reviewed. The significant works in DAI along organisational line were listed. Analysing the existing researches in DAI with a comparison to organisation theory, the shortcomings were identified as follows:

1. The notion of origination is subordinate to the agent. It is an attribute of a rational agent when it is engaged in a social environment.

2. The performance of a multi-agent system is measured by the performance of individual agents.

3. Most cooperation frameworks and models are focused on one *import-conversion-export* process.

4. Organisational arrangement for a multi-agent system copies human organisational structures and is applied as an external constraint to its agents. Agents in the system are only role players rather than organisation constructors.

However, many useful models focused on one process of organisational goal attainment and can be used to construct a more complete description of an organisational goal attainment process.

The SMM framework presented in section 5.3 is a proposal of a more complete description of an organisational goal attainment process. It is based on organisation theory and research in DAI. Its six-stage framework consists of goal selection, act selection, team formation, plan formation, team action and shifting. Each stage is a different kind of *import-conversion-export* process and may have many processes.

Providing a framework is not the ultimate goal of this research. Theoretical support is needed for the framework and to formalise the framework for both application and formal proving. Allocating the existing theory in support of the SMM framework, this thesis has found the need for a decision theory for agent's mental state and act shifting. It can be used to answer the question *why joint activities are formed* and *to what extent an agent may abandon it efforts towards joint activities.*

Chapter 6

A Quantitative Decision Theory

Whenever I have studied human affairs, I have carefully laboured not to mock, lament, or condemn, but only to understand.

-- Spinoza

This chapter, as the second part of the theoretical contribution of this thesis, demonstrates what needs to be considered when an agent makes a decision between alternative actions, α and α', to achieve a goal φ. It is used to answer the fundamental question of why joint activities are formed. There are two questions: 1) *why an agent chooses to act in a team as an alternative to working in isolation or in another team.* 2) *To what extent should an agent change its mind and search for other alternatives in case the teamwork attempt fails*?

The contents in this chapter are arranged in the following order. The foundations of the

theory are introduced in section 6.1 to draw a clear boundary for the theory. The *preference* as the central notion of an agent's mental states for decision-making is then introduced in section 6.2. In section 6.3, a single continuous quantity variable named *expected utility* is introduced to pave the way for a decision measurement. Finally section 6.4 provides a way of measuring the elements which are involved in decision-making based on the developed theory.

6.1 Foundations of the Theory

When developing a decision theory on actions for agents, a number of issues need to be resolved. The first issue is the concern about the decision-maker that is the definition and the representation of an individual agent. The second issue regards the subject of the decision-making that is 'action' in this thesis' concern. The actions, which are talking about, are different but can achieve the same goal. The third issue reflects the decision process including: the elements that needed to be considered; the measurement of the elements that have been considered; the strategies that are applied in the decision-making, and so forth. The third issue regards the origin of the theory. The location of the origin can be in practice (such as operation research, cost-efficiency analysis and system analysis) or in philosophy (more general study based on feasibility and desirability). Resolving these issues provides a foundation for the theory.

The following foundations are applied in this thesis:

1. **The agent**. Agents are autonomous and intelligent. The autonomy enables agents to decide, choose and act on their own knowledge. Intelligence will prevent them doing any thing against the criteria that have been set up by human agents. The integration of autonomy and intelligence can be viewed as rational. Therefore, our agents are rational agents.

2. **The origin and the scope of the theory**. A rational agent should be able to act in a particular way that pleases the human agents. One reason for this is that the rationality is a set of criteria which have been set up by human agents although the human agents are subject to what is called "bounded rationality" [Simon 57, 78]. So an action is rational if it is judged so by human agents. Another reason for this is that the ultimate goal of developing agents is to build a tool to work for human agents. Actually many researchers view agents as *artificial life* [Longton 93, 94]. If agents are so well developed (autonomous and intelligent) they should adopt any human rational behaviour. Therefore, our theory is developing based on human rational behaviour. Studying human rational behaviour particularly on decision-making, our focus is not on any particular practice such as engineering, economy and so forth, instead it is originated on a fundamental and philosophical human decision process. Finding out the elements, which are involved in, and the impact of these elements on the action decision, the two axes of our study are feasibility and desirability.

3. **The divisibility of action.** An action, as the objective of our study in act selection, must be *divisible*. This divisibility permits the usage of a continuous real variable to express complex results of an action such as decision-making. It also provides a possibility for a quantitative measurement of the variable to be developed.

In a more general sense, the behaviour of a rational agent can be viewed as the *best* use of limited alternatives. It is the notion of *best* that causes problems. Different disciplines have their own subject of study and searching for the best use of limited alternatives, such as military, political and economic science, all have a different interpretation of the best usage. However, in practice a particular discipline can always find its clearest expression. For example, economists confine their study

particularly to the use of limited alternatives in producing and consuming goods and services (assumed to be *divisible* into small physical units). Market prices (the continuous real variable) of goods and services are used to express the complex results of many *actions* in the form of monetary profit (a single continuous quantity).

It so happens, however, that our concern with the *best use of available alternatives* regarding an agent's chosen action also finds its clearest expression when the alternative actions are treated as divisible entities.

Based on the above foundations action decision theory can now be developed.

6.2 The Theory of Decision

In this section the theory of decision is developed by introducing a notion of *preference* as a mental state of decision-making agent. To react to the environment, in which the decision is made, the notion of preference is further expressed by another two notions *taste* and *belief*. The remainder of this section will exploit these two notions and their consistency that is needed to ensure that the decision made is consistent.

Generally, a decision is an action of choosing among available alternatives. This action is controlled by a mental state called *preference*. Notion of preference, denoted[1] by (Prefer i α α'), states an agent i prefers action α to any other action α' in any *possible world*[2]. If agent i decides to adopt action α to achieve goal φ, it means among all alternative actions that can achieve goal φ, agent i prefer action α to any other alternative actions. Formally, (Decide i α φ) is used to express an agent i deciding to

[1] The symbols used in this thesis adopt the symbolic system used in the most DAI theoretic works. They mostly have straightforward meanings. However wherever possible, an explanation always provided. A complete symbol system is provided in Appendix B.

[2] Possible world is a classical model of reasoning [Reeves and Clarke 90]. The basic idea is that there are a number of other worlds (states of affair) except the actual world and they are logically possible and not contradict with the actual world. The details are described in Chapter 7 as a logical preliminary for formalising the SMM framework.

adopt action α to achieve goal φ. Then the following expression can be defined[3].

(6.1) $$(\text{Decide } i\ \alpha\ \varphi) \stackrel{def}{=} (\text{Goal } i\ \varphi) \wedge (\text{Achieve } \alpha\ \varphi) \wedge (\text{Prefer } i\ \alpha\ \alpha').$$

The mental state preference is embodied by two distinct basic mental states namely *tastes* and *beliefs*: tastes describes a mental state that an agent has, a particularly order of alternative actions[4]; beliefs describes the mental state of an agent that is a particular order of future alternative events according to its views of their comparative *probabilities*. The tastes and beliefs of a rational agent have to be consistent. This consistency will be defined later in the section.

6.2.1 Decision under Certainty - Tastes and Its Consistency

Formula 6.1 shown that an agent i decides upon adopting an action α to realise its goal φ because the agent prefers the action α to any other alternative actions. Under certainty, there are no factors that can not be controlled by the decision-making agent. In this case, an agent's mental state of preference depends only on its tastes because of the certainty. Therefore an agent's decision to choose amongst available alternatives is a simple one in the sense that the actions and their results are coincident: each can be identified with the alternative chosen.

Let us study an action and its result in more detail. In a general case, it is useful to distinguish between an action α and its result. Let

(6.2) $$r = \rho(\alpha),$$

---

[3] Here α' represents all the conceivable actions except α. This is clear by using different symbols α and α'. One may argue that α' should be specified being able to achieve goal φ. It will be clear later when distinguishing between conceivable actions and feasible action is provided.

[4] Under certainty the order of taste is order of preference. Under uncertainty, taste refers to the order of actions according to their outcomes.

ρ is called the outcome function. Thus α_1 and α_2 may be two actions, and $r_1 = \rho(\alpha_1)$, $r_2 = \rho(\alpha_2)$ are the respective outcomes. If r_1 and r_2 are available and can be evaluated for choice, then a rational agent will choose the more valuable outcomes. But in our consideration, it is the actions, not the outcomes that are chosen. However, it will prove useful to define preference ordering on outcomes as a special case of preference ordering on actions. Therefore when an agent prefers α_1 to α_2, in this work the agent prefers $\rho(\alpha_1)$ to $\rho(\alpha_2)$. i.e.,

(6.3) (Prefer i α_1 α_2) $\stackrel{def}{=}$ (Prefer i r_1 r_2), if and only if, $r_1 = \rho(\alpha_1)$ and $r_2 = \rho(\alpha_2)$.

Empirically the tastes of a rational agent have to be consistent. That means if the criteria of what an agent considers "valuable" are not in some sense fixed, making its decision "consistent," it would not be ascertainable whether a preference is or is not made of available alternatives and the word "rational" would therefore be meaningless. Consistency of tastes is defined in the following way:

Definition 1 (Consistency of Tastes): An agent's tastes are consistent if and only if the following are satisfied.

1) **Prefer.** An agent i prefers α_1 to α_2, if it never chooses α_2 when α_1 is available.

2) **Indifferent.** An agent i is indifferent between α_1 and α_2, if from sets of alternatives containing both α_1 and α_2 the agent sometimes chooses α_1, and sometimes α_2.

3) **Complete.** An agent i is either indifferent between any α_1 and α_2 or it prefers one to the other.

4) **Transitive.** An agent i prefers α_1 to α_2, and α_2 to α_3, it therefore prefers α_1 to α_3.

6.2.2 Decision under Uncertainty - Beliefs, Tastes and Their Consistency

In the previous section, decision under certainty is an ideal situation. However, in practice the environment is one which involves uncertain factors that means decision-maker can only hope, predicate and expect, anything rather than certainty. This section will study the decision theory in such an environment.

1. Environment and uncertainty

The same action can result in different outcomes, depending on factors that are not controlled by the decision-making agent. For example, if an agent is attempting to organise a team, its efforts are not guaranteed to achieve the result of team formation because the factors, such as the mental states of other agents, are not controlled by the organising agent. These factors can be denoted by a variable, x, called environment or external world. Thus the outcome function 6.2 can be rewritten into

$$r = \rho(x, \alpha). \tag{6.4}$$

In general, the value of x is unknown to an agent in advance, and therefore the result r of an action α is not known even if the outcome function ρ is known. This case is regarded as the uncertainty is existing about the variable x, and therefore also about the variable r, in a given action α.

For example, an agent is attempting to persuade another agent to work together as a team, but it is uncertain about to what extent it needs to persist in this attempt because of uncertainty about the other agent's current attitude and future contribution. If α denotes the effort of the attempt, x the expected contribution from the other agent, then the resulting profit to the agent's attempt is given by the outcome function

$$\rho(x, \alpha) = x - \alpha.$$

Here the action variable is the quantity of the attempt α, and the state of the environment is described by the expected contribution x (from other agents).

What if there is uncertainty about the outcome function, ρ, itself, as well as about the state of the environment? This difficulty can always be overcome by describing the *possible world* of the environment in sufficiently great detail. With reference again to the agent's team organising example, the agent may not know which of several quantifying functions will actually apply to the attempt. For example, the agent may evaluate the attempt with different functions c_1, c_2 and so forth, then the state of the environment can be described as the contribution of the agent's opponent, by p (profits) and the number n corresponding to the quantifying function which the agent adopted. The appropriate outcome function is now

$$\rho(p, n; \alpha) = p - c_n(\alpha);$$

the form of the function is once again certain, although the variable p and n describing the state of the environment are not. The environment variable x is now a vector: $x = (p, n)$.

In general, symbol X is used to denote the set of all possible states of the environment, and R is used to denote the set of results. Therefore every action α determines a function, say $\mathbf{f}_\alpha$, from X to R, namely,

$$\mathbf{f}_\alpha(x) = \rho(x, \alpha).$$

Conversely, every function $\mathbf{f}$ from X to R can be thought of as being generated by an action, $\alpha_\mathbf{f}$, with the outcome function defined by

$$\rho(x, \alpha_\mathbf{f}) = \mathbf{f}(x).$$

A function **f** from X to R is called an act. Dealing with acts, rather than with *actions* and *an outcome's function*, has the advantages of focusing attention on the important point that the essential thing to know about an action is not its *name*, but its consequences under alternative states of the environment. The concept of *act* does help to define what it meant by *the set of all conceivable actions*; by this it shall mean either the set of all acts or some suitable index set for the set of all acts. This set will be denoted by A. Not all *conceivable* actions are feasible in a given decision situation, such as a robot choosing from alternative paths to move a block from one location to another location. In this case, not all the possible paths are executable either because of the obstacles in the paths or a robot's physical constraints on its joints. Therefore a *feasible subset* of A needs to be indicated.

With regards to a decision under uncertainty, its preference can be characterised by a set X of alternative states x, a set R of alternative outcomes r, the set A of conceivable actions α, and an outcome function ρ from $X \times A$ to R, which specifies the outcome resulting from each state-action pair,

$$r = \rho(x, \alpha). \tag{6.5}$$

2. Consistency of beliefs and tastes

After uncertainty is introduced, an agent's preference and the consistency of preference become more complex than they are under certainty. Formally, it is possible to simply maintain the previous definition of preference and the consistency of preference under certainty even after introducing uncertainty. One might simply keep an agent's tastes, a particular preference ordering on actions, and not inquire into the reasons underlying this ordering. For example, the model of CPS introduced in Chapter 4 does not inquire into the reason for an agent preferring to work with other agents as part of a team rather

than working in isolation. It will not suffice for the purposes of explaining why joint activities between agents are formed.

To study the concrete problem of preference under uncertainty it is useful to constrain an agent's preference to the following principles of consistent preference:

Definition 2 (The Principles of Consistent Preference):

1) Completeness: The ordering of actions according to preference must be complete.

2) Independence: The two elements of preference, tastes and beliefs, of an agent must be independent.

The above preference consistent principles emphasise the following four points:

1) Not only consistent tastes, but also consistent beliefs need to be defined.

2) An agent's preference should be free of what it may be termed as "wishful thinking", that is the agent's ranking of the probabilities of events should not depend on whether a given event will result in a more or less desirable outcome of a given action.

3) It is also necessary that an agent's preferences among actions under uncertainty and its preferences among outcomes under certainty should be consistent.

4) An agent's preference should obey the social norms and rules of logic.

6.3 Quantisation of Preference - Expected Utility

In order to study the consistent preference of an agent under uncertainty, and to provide a simple way to measure an agent's preference, a single continuous quantity variable

will be introduced in this section. This single continuous quantity variable is called *expected utility*[5]. With it the consistent preference can be defined. This consistent preference actually implies a *higher expected utility*.

In this approach, the tastes as ranks of alternative outcomes are replaced by numbers called *utilities*; and the probability ranks of alternative *events* are replaced by numbers called *subjective probabilities*. These numbers retain the same respective orderings as the ranks they have replaced; but unlike mere ranks, utilities and subjective probabilities can be meaningfully added and multiplied. Moreover, subjective probabilities obey the usual rules of the probability calculus. Therefore, the expected utility of an action under uncertainty can be defined as:

Definition 3 (Expected Utility):

> The expected utility of an action under uncertainty is the average of the utilities of its several possible outcomes, each weighted by the probability of the event under which that outcome will obtain.

In the above definition, the expected utility of an action under certainty is then identical with the utility of its outcome. The preference ordering of actions is identical to the ordering of actions by expected utilities.

To formalise the expected utility, it is important to understand the meaning of the *event* in the above definition of the subjective probabilities. The event means any set of states, that is, any subset of X. Suppose that the set R of alternative outcomes is finite, so R consists of the N alternative outcomes $r_1, r_2, ..., r_N$. For any action α, let $\sigma_i(\alpha)$ denote the set of states x such that

(6.6) $$\rho(x, \alpha) = r_i;$$

[5] The *expected utility* is a term borrowed from the Social Science [Fishburn 68].

in other words, $\sigma_i(\alpha)$ is the event *action* α *has outcome* r_i. Normally, it is called event

$\sigma = \{x_i\}$, where $x_i \in X$, $1 \leq i \leq n$ and $\rho(x_i, \alpha) = r_i$.

The *expected utility* denoted by $\Omega(\alpha)$ for a given action α is then expressed by

$$\Omega(\alpha) \stackrel{def}{=} \frac{1}{N} \sum_{i=1}^{N} \upsilon(r_i)\, \pi[\sigma_i(\alpha)]. \tag{6.7}$$

Where υ and π are two "auxiliary" numerical functions: function π on the events, called *subjective probability function*, and function υ on R, called the *utility function*. Therefore, the preference (Prefer $i\ \alpha\ \alpha'$) can be redefined as

$$(\text{Prefer } i\ \alpha\ \alpha') \stackrel{def}{=} \Omega(\alpha) \geq \Omega(\alpha'). \tag{6.8}$$

That is, a preference ordering can thus be represented, explained by the *expected utility*.

The given expected utility function $\Omega(\alpha)$ in (6.7) shows that the expected utility of an action α does not only depend on the action α itself, but also the given functions ρ, π, and υ. These functions summarise the factors beyond a mere agent's control: its beliefs π on the events, its tastes υ on the outcomes, and its idea of the "physical" relation ρ that states how outcomes are determined by itself and by the environment.

Return to the initial purpose of introducing the notion of preference, which is to study the possibility to express and explain an agent's decision in a measurable manner. Having defined expected utility as the single continuous quantity variable, the objectives of the remainder of the chapter thereafter are to show that:

1. under certain *plausible conditions*, the expected utility ordering can be represented, and explained by the probability and the utility functions. These *plausible conditions* are actually a set of natural rules, which reveal the *independence*

between tastes and beliefs. In other words, only by proving the independence between tastes and beliefs, can the expected utility ordering be represented by the probability and the utility functions. This is what is to be explained in the remainder of this section.

2. Under the conditions of independence between tastes and beliefs, the probability and the utility functions can be measured in any particular case. A possible way of the measurement will be provided. This is what is to be introduced in section 6.4.

6.3.1 Independence Between the Tastes and the Beliefs

The requirement for independence between an agent's tastes and its beliefs has already been mentioned. Economic decision theory reveals three distinguishing aspects of independence between an agent's tastes and beliefs [Marschak and Radner 72]. They have been adopted and listed below. These aspects can be associated, respectively, with the possibility of defining:

1. Conditional preferences among actions in the given events.
2. An ordering of outcomes according to preferences, independent of the states in which they occur.
3. An ordering of states according to probability, independent of the outcomes with which they are associated.

Independence condition 1: Conditional preferences in one event are independent of consequences in other events.

Consider a given event σ, and any two actions α_1 and α_2, that result in the same consequences outside σ. That is

(6.9) $$\rho(x, \alpha_1) = \rho(x, \alpha_2), \quad \text{for } x \text{ not in } \sigma,$$

which means when σ does not happen the outcomes of both actions α_1 and α_2 are the same. The first independence condition states that *the preference ordering of two such actions should be independent of their (common) outcomes outside* σ.

Formally, let α_1' and α_2' be two other actions such that

(6.10) $$\left.\begin{aligned}\rho(x, \alpha_1') &= \rho(x, \alpha_1)\\ \rho(x, \alpha_2') &= \rho(x, \alpha_2)\end{aligned}\right\} \text{ for } x \in \sigma,$$

$$\rho(x, \alpha_1') = \rho(x, \alpha_2') \quad \text{for } x \notin \sigma.$$

The pairs (α_1, α_2) and (α_1', α_2') will now be considered separately, each as a different feasible subset of the set A of all conceivable actions. The symbol $\underset{A}{\preceq}$ is denoted to represent the preferences among actions, therefore

(6.11) $$\alpha_1 \underset{A}{\preceq} \alpha_2, \quad \text{if and only if,} \quad \alpha_1' \underset{A}{\preceq} \alpha_2'.$$

Notice that α_1' and α_2' can be thought of as obtained from α_1 and α_2 by modifying their common outcomes outside σ. Independence condition 1 states that, as long as these outcomes remain common to both actions, such modifications should not affect the choice between actions. It is a rather natural sense in human preference on two actions. We concern two actions always relating to a particular event rather than other possible events out side the concern event.

Independence condition 1 makes it possible to define conditional preferences among actions. Consider any two actions, α_1 and α_2, and any event σ; construct two other actions α_1' and α_2' such that if σ happens, then α_1' has the same outcomes as α_1, and α_2' has the same outcomes as α_2; but, if σ does not happen, then α_1' and α_2' have common outcomes:

(6.12)
$$\left.\begin{array}{l}\rho(x, \alpha_1) = \rho(x, \alpha_1') \\ \rho(x, \alpha_2) = \rho(x, \alpha_2')\end{array}\right\} \quad \text{for } x \in \sigma,$$

$$\rho(x, \alpha_1') = \rho(x, \alpha_2') \qquad \text{for } x \notin \sigma.$$

define

$$\alpha_1 \underset{A}{\preceq} \alpha_2 \qquad \text{given } \sigma, \text{ to mean} \qquad \alpha_1' \underset{A}{\preceq} \alpha_2'.$$

Independence condition 2: Tastes are independent of beliefs. In another words, the conditional ordering of outcomes given $\{x\}$ is independent of x.

In this case, an event σ consists of a single state x. Thus $\sigma = \{x\}$. The conditional ordering of actions given the event $\{x\}$ defines a conditional ordering of outcomes r given the event $\{x\}$, since an action, as a function from the set of states to the set of outcomes, results in a unique outcome for any given state. The conditional ordering of outcome r_1 and r_2 given $\{x\}$ denoted by $\underset{R}{\preceq}$ can be written

$$r_1 \underset{R}{\preceq} r_2 \qquad \text{given } \{x\}$$

whenever $r_1 = \rho(x, \alpha_1)$, $r_2 = \rho(x, \alpha_2)$ and

$$\alpha_1 \underset{A}{\preceq} \alpha_2 \qquad \text{given } \{x\}.$$

The second independence condition states that these conditional orderings of outcomes should be identical.

Independence condition 3: Beliefs are independent of tastes.

To illustrate the meaning of the independent condition 3, consider an example, where an agent i that represents robot1 who must choose between two actions. Action α consists of working with agent j which represents robot2, as a team, which is only feasible when *agent j is willing to work with it as a team* (event W); action α^* is to work alone, which

happens only *when agent j does not work with agent i* either agent *j* does not want to or it is unable to work with *i* (event non-*W*, denoted by $\overline{W}$). Suppose the success of either action is equally desirable: for example, let success mean in both cases the goal can be achieved, here for clarity, an explicit amount of benefit is used to represent the achievement of the goal. Similarly, let the desirability of failure of either action be the same, failure consisting in both cases of a certain small amount cost, as in Table 6.1.

Table 6.1 Benefits and Costs of Actions

Events / Actions	W	$\overline{W}$
α	1000	-20
α^*	-20	1000

It is consistent with ordinary usage to say that the agent *i*'s comparison of probabilities of the two events is revealed by its preference ordering of the two actions. For, if agent *i* *prefers* to work with agent *j* as a team (action α), we usually say that agent *i* believes that agent *j*'s willingness to work with it as a team (event *W*) to be more *probable* than agent *j* does not (event $\overline{W}$). More generally, let *Y* and $\overline{Y}$, *Z* and $\overline{Z}$, be another two pair of exhaustive and mutually exclusive events. Let α_1 and α_2 denote other two actions that will yield the same benefits and costs as action α does. They are illustrated in Table 6.2.

Table 6.2 Preferences among multi-event with same action outcomes

	W	$\overline{W}$		Y	$\overline{Y}$		Z	$\overline{Z}$
α	1000	-20	α_1	1000	-20	α_2	1000	-20

When an agent *i* prefers action α to action α_1, it is usual to say that, to agent *i*, *W* seems to be more probable than *Y*. Now if agent *i* prefers α to α_1 and α_1 to α_2, and is consistent, then agent *i* will prefer α to α_2. Hence, if, to a consistent agent, *W* seems more probable than *Y*, and *Y* than *Z*; then to this agent, *W* seems more probable than *Z*.

The ordering of the probabilities of these three events having been defined, so far, with respect to a particular pair of outcomes such that (1000, -20). Thus the transitivity of preferences among actions has induced the transitivity of probabilities of events with respect to the particular pair of outcomes. It can be generalised that the ordering of probabilities on events to different outcomes. To the same agent i, the ordering of the event's probabilities is that W is more probable than Y, and Y than Z when the pair of outcomes is (1000, -20) will, if this agent is consistent, preserve the same ordering among the probable events when the pair of outcomes is (s, f), where s ("success") is something, anything, that the agent prefers to f ("failure").

The independence condition 3 states a consistent agent's judgement about comparative probabilities of events should not depend on what rewards or punishments they entail. In Table 6.3, if an agent prefers α to α_1 to α_2, it should prefer α' to α_1' to α_2'.

Table 6.3 Preferences among multi-event with same action outcomes

	W	$\overline{W}$		Y	$\overline{Y}$		Z	$\overline{Z}$
α	1000	-20	α_1	1000	-20	α_2	1000	-20
α'	s	f	α_1'	s	f	α_2'	s	f

Formally, Let f, s, f', s' be outcomes such that

$$f \underset{R}{\prec} s^{6}, \qquad f' \underset{R}{\prec} s'.$$

Let σ and σ' be two events, and define actions α_1, α_2, α_1', α_2' by

(6.13)
$$\rho(x, \alpha_1) = \begin{cases} s & if\ x \in \sigma \\ f & if\ x \notin \sigma \end{cases} \qquad \rho(x, \alpha_2) = \begin{cases} s & if\ x \in \sigma' \\ f & if\ x \notin \sigma' \end{cases}$$

$$\rho(x, \alpha_1') = \begin{cases} s' & if\ x \in \sigma \\ f' & if\ x \notin \sigma \end{cases} \qquad \rho(x, \alpha_2') = \begin{cases} s' & if\ x \in \sigma' \\ f' & if\ x \notin \sigma' \end{cases}$$

[6] Symbol "$\prec$" means restrict less preferable. It will be introduced later in the chapter. Because "failure" is normally less preferable then "success" in our consideration, "$\prec$" is more suitable than "$\preceq$".

then $\alpha_1 \underset{A}{\preceq} \alpha_2$ if and only if $\alpha_1' \underset{A}{\preceq} \alpha_2'$.

In the situation of (6.13), if $\alpha_1 \underset{A}{\preceq} \alpha_2$, it is usual to say that event σ is not more probable than event σ', and the symbol $\underset{X}{\preceq}$ is denoted to represent the subjective probability ordering of events, thus,

(6.14) $\sigma \underset{X}{\preceq} \sigma'$.

6.3.2 The Sure-Thing Theorem

Recall the consistency of an agent's tastes defined in 6.1.3 where preference, indifference and completeness are simple under certainty. Under uncertainty equivalent definitions are needed. However, instead of defining them, a theorem called Sure-Thing Theorem[7] is given based on the three independence conditions introduced in the previous section.

First, the two new preference orders and notations need to be introduced. "Equivalent" denoted by "$\sim$" and "strictly preferred" denoted by "$\prec$" are defined as follows,

$\alpha_1 \sim \alpha_2$ if and only if $\alpha_1 \preceq \alpha_2$ and $\alpha_2 \preceq \alpha_1$,

$\alpha_1 \prec \alpha_2$ if and only if $\alpha_1 \preceq \alpha_2$ but not $\alpha_2 \preceq \alpha_1$.

Second, *partition* of X means a collection of events $\sigma_1, ..., \sigma_n$ such that

I. every state x is in some event σ_i; and

II. no state x is in two *different* events σ_j and σ_k.

[7] Sure-Thing Theorem is first introduced in [Savage 54] and elaborated by [Marschak and Radner 72] for an economic decision theory. This thesis extends it into agent's decision-making under uncertainty.

Theorem 1 (Sure-Thing): Let $\{\sigma_i\}$ $i = 1, ..., n$ be a partition of X, and let α_1 and α_2 be two actions.

I. If, for every i, $\alpha_1 \underset{A}{\preceq} \alpha_2$ given σ_i, then $\alpha_1 \underset{A}{\preceq} \alpha_2$.

II. If, further, $\alpha_1 \underset{A}{\prec} \alpha_2$ given σ_j, for some j, then $\alpha_1 \prec \alpha_2$.

III. If , for every i, $\alpha_1 \underset{A}{\sim} \alpha_2$ given σ_i, then $\alpha_1 \underset{A}{\sim} \alpha_2$.

The above *sure-thing* theorem states the most safe preferences among actions under uncertainty (possible events and the possibility of an events' occurrence).

Proof: Suppose X can be partitioned into n events $\sigma_1, ..., \sigma_n$. Any two events σ_j and σ_k among the n events have four states x_1, x_2, x_3, x_4, where $\sigma_j = \{x_1\ x_2\}$ and $\sigma_k = \{x_3\ x_4\}$.

i). The first condition in Sure-Thing Theorem, two actions α_1 and α_2, have $\alpha_1 \underset{A}{\preceq} \alpha_2$ in every given σ_i, thus $\alpha_1 \underset{A}{\preceq} \alpha_2$ given σ_j and $\alpha_1 \underset{A}{\preceq} \alpha_2$ in given σ_k. Now constructing a third action α_3 which is in between α_1 and α_2, with respect to preference, as displayed in Table 6.4.

Table 6.4 Illustration of action, states and events

Events / States / Actions	σ_j		σ_j	
	x_1	x_2	x_3	x_4
α_1	r_1	r_2	r_3	r_4
α_3	s_1	s_2	r_3	r_4
α_2	s_1	s_2	s_3	s_4

By definition of conditional preference (independence condition 1), $\alpha_1 \underset{A}{\preceq} \alpha_3$, since $\alpha_1 \underset{A}{\preceq} \alpha_2$ given σ_j. Similarly, $\alpha_3 \underset{A}{\preceq} \alpha_2$ since $\alpha_1 \underset{A}{\preceq} \alpha_2$ given σ_k. Therefore, by the transitivity of preferences, $\alpha_1 \underset{A}{\preceq} \alpha_2$.

ii). From the definition of "strict preference", "$\alpha_1 \prec \alpha_2$" means "$\alpha_1 \underset{A}{\preceq} \alpha_2$" and not "$\alpha_2 \underset{A}{\preceq} \alpha_1$". The first condition of Sure-Thing theorem that states the first part of the condition "$\alpha_1 \underset{A}{\preceq} \alpha_2$", and the second condition not "$\alpha_2 \underset{A}{\preceq} \alpha_1$" can also be easily deducted from the first condition of the Sure-Thing Theorem. Not $\alpha_2 \underset{A}{\preceq} \alpha_1$ means not "every σ_i that $\alpha_2 \underset{A}{\preceq} \alpha_1$", further it means there is some j, where $\alpha_2 \succ \alpha_1$, i.e., $\alpha_1 \prec \alpha_2$ given σ_j. It is exactly the second condition of the Sure-Thing Theorem.

iii). For every i, $\alpha_1 \underset{A}{\sim} \alpha_2$ given σ_i, according to the definition of equivalence "~", $\alpha_1 \underset{A}{\preceq} \alpha_2$ in every given σ_i and $\alpha_2 \underset{A}{\preceq} \alpha_1$ in every given σ_i. According to the first condition of the Sure-Thing Theorem, $\alpha_1 \underset{A}{\preceq} \alpha_2$ and $\alpha_2 \underset{A}{\preceq} \alpha_1$. That means $\alpha_1 \underset{A}{\sim} \alpha_2$. ***End of Proof.***

Lemma (Sure-Thing for Outcomes): If for every state x in X,

$$\rho(x, \alpha_1) \underset{R}{\preceq} \rho(x, \alpha_2), \text{ then } \alpha_1 \underset{A}{\preceq} \alpha_2.$$

This lemma is obvious from the independence condition 2 and the Sure-Thing theorem. The proof is ignored here. The point made by this lemma and together with the

independence condition 2 tell us, in effect, that the ordering of outcomes is identical with the ordering given by conditional preferences.

6.4 Measurement of the Preference

As stated earlier in the previous section, the objective of this section as the last part of the chapter will provide a way of measuring preference. In the above section, the order of preference on actions by a rational agent is identical with the order of the expected utility on actions (formula 6.8). Therefore a measure of preference can be carried out by measuring the expected utility. The expected utility of an action under uncertainty is the average of the utilities weighted by the probability of the event under which that outcome will be obtained (formula 6.7). The independence between tastes and beliefs, together with the Sure-Thing Theorem described in the previous section enable that a measurement of the expected utility can be realised by measuring of the subjective probabilities and the utilities.

6.4.1 The Properties and the Measurement of Subjective Probabilities π

The independence of tastes and beliefs permits an agent's ordering of events to be influenced only by probability and not the particular outcomes of any actions. Conventional probability theory and calculus can be applied to an agent's subjective probability function π in the expected utility function $\Omega(\alpha)$.

1. Property of subjective probabilities

In general, for a given positive integer n, the possible alternative states X can be partitioned into n equiprobable events $\sigma_1, ..., \sigma_n$. Let $\mathcal{W}'$ denote the class of all events (sets) of the form

(6.15) $$(\sigma_{i_1} \text{ or } \sigma_{i_2} \text{ or } \ldots \text{ or } \sigma_{i_m}),$$

where $i_1, i_2, \ldots, i_m$ are distinct integers between 1 and n. Each event σ of the form (6.15) has been assigned the numerical *subjective probability* $\pi(\sigma) = m/n$. This function π represents the ordering of events in $\mathcal{W}'$ according to subjective probability, in the sense that for σ, σ' in $\mathcal{W}'$, then

(6.16) $$\sigma \underset{X}{\preceq} \sigma' \qquad \text{if and only if} \quad \pi(\sigma) \leq \pi(\sigma').$$

The function π satisfies the following properties:

1. $\pi(\sigma) \geq 0$, for all $\sigma \in \mathcal{W}'$

(6.17) 2. $\pi(X) = \sum_{i=1}^{n} \pi(\sigma_i) = 1$

3. $\pi(\sigma \text{ or } \sigma') = \pi(\sigma) + \pi(\sigma')$, $\sigma, \sigma' \in \mathcal{W}'$ and $\sigma \cap \sigma' = \varnothing$.

Thus by (6.17), the function π has all the properties of a probability measure for those events belonging to $\mathcal{W}'$. However, besides the basic properties of the subjective probability function π from the conventional theory of probability described above, the concept of *conditional probability* and *independence* can also be developed. In particular, the conditional probability of σ given σ' is defined by

(6.18) $$\text{Prob}[\sigma \mid \sigma'] \equiv \frac{\pi(\sigma \text{ and } \sigma')}{\pi(\sigma')},$$

provided that $\pi(\sigma') > 0$. The sets σ and σ' are said to be independent if

(6.19) $$\pi(\sigma \text{ and } \sigma') = \pi(\sigma)\, \pi(\sigma').$$

This implies

(6.20) $$\text{Prob}[\sigma \mid \sigma'] = \pi(\sigma), \quad \text{if } \sigma \text{ and } \sigma' \text{ are independent.}$$

2. Measurement of subjective Probabilities

To explain the measurement of the subjective probability, consider an arbitrarily large integer n that X can be partitioned into n equiprobable events. To be precise, suppose that such a partition can be constructed for each number n_i in some increasing, unbounded sequence n_1, n_2, and so forth, of positive integers. Then, for each n_i, a class $\mathcal{W}_i$ of events and a probability measure π_i can be constructed satisfying (6.16) and (6.17). Suppose, further, that if $n_i \geq n_j$, then $\mathcal{W}_i$ contains $\mathcal{W}_j$, that is every event in $\mathcal{W}_j$ is also in $\mathcal{W}_i$. This could be accomplished, in particular, if the partitions in question could be constructed by a process of successive subdivisions, for example, by dividing each event in any one partition into two equally probable events, to yield the succeeding partition. Still using the example of the robot-agent example, agent i which represents robot1 is choosing between working with agent j, which represents robot2 as a team or working alone, the partition construction could be done by exploiting the continuity of a variable "willingness" of working with agent i of agent j, which is assessed and used by agent i. In this case, the sequence of numbers n_i would be 2, 4, 8, and so forth.

If each class $\mathcal{W}_i$ includes the preceding one, as in the above construction, then it is obvious that all the probability measures π_i agree, in the sense that if an event σ belongs to both $\mathcal{W}_i$ and $\mathcal{W}_j$, then

(6.21) $$\pi_i(\sigma) = \pi_j(\sigma).$$

Generally, for a class $\mathcal{W}$ of all events σ that are in class $\mathcal{W}_i$ for some i, its probability function π has the properties of (6.16) and (6.17), with $\mathcal{W}'$ replaced by $\mathcal{W}$. Let $\mathcal{R}$ be a set of rational numbers of the form (m/n_i), $0 \leq m \leq n_i$; then:

(6.22) 1. For every event σ, $\sigma \in \mathcal{W}$, $\pi(\sigma)$ is a rational number and $\pi(\sigma) \in \mathcal{R}$

2. For every rational number $m/n \in \mathcal{R}$, there is an event σ, $\sigma \in \mathcal{W}$ such that

$\pi(\sigma) = m/n.$

Therefore, it is possible for an agent to assign every rational number k in $\mathcal{R}$ to event σ_k with probability k. Now, these events σ_k can be used as "bench marks" to measure the probabilities of other events outside the class $\mathcal{W}$.

Theorem 2 (Measurement of event's probability): The probability of any event σ can be measured to any desired degree of accuracy by using the bench mark events σ_k. i.e. for a given set of rational number set $\mathcal{R}$, there are some rational number k in $\mathcal{R}$, and

(6.23) $\sigma \underset{X}{\sim} \sigma_k$ then

$$\pi(\sigma) \equiv k.$$

***Constructive proof*:**

In this proof, a constructive method is illustrated by employing successive approximations that are based on the fact that every number between 0 and 1 can be approximated as well as one pleases by a number in $\mathcal{R}$. Suppose that $\mathcal{R}$ is the set of all rational numbers of the form $(m/2^p)$, $p \geq 1$, $0 \leq m \leq 2^p$. One can start by asking whether

$$\sigma \underset{X}{\prec} \sigma_{1/2};$$

if the answer is "yes," then ask whether

$$\sigma \underset{X}{\prec} \sigma_{1/4};$$

if the answer is "no," then ask whether

$$\sigma \underset{X}{\prec} \sigma_{3/8};$$

and so forth. In this way a sequence of approximations can be construct, so that after N steps one can make a statement of the form

$$\sigma_{\frac{M_N}{2^N}} \underset{X}{\prec} \sigma \underset{X}{\prec} \sigma_{\frac{M_N+1}{2^N}} \qquad 0 \le M_N \le 2^N;$$

furthermore, by construction, the sequence $(M_N/2^N)$ is non-decreasing, and the sequence $(M_N + 1)/2^N$ is non-increasing. In the above illustration, the sequence $(M_N/2^N)$ may start out 0, 1/4, 1/4, and the sequence $(M_N + 1)/2^N$ may start out 1/2, 1/2, 3/8. Therefore the subjective probability of event σ can be defined as

(6.24) $$\pi(\sigma) = \lim_{N\to\infty} \frac{M_N}{2^N} .$$

Formula (6.23) is a special case of formula (6.24), since all real numbers between 0 and 1, rational or otherwise, can be approximated to any desired degree of accuracy by numbers of the form $(M/2^N)$, with $0 \le M \le 2^N$.

End of proof.

6.4.2 The Properties and Measurement of the Utility Function υ

Similar to the description of the subjective probability function π, the properties of the utility function υ are studied to ensure that expected utility function Ω represents the preference order $\underset{A}{\prec}$, and then how the utility function can be measured is illustrated.

1. Property of utility function υ

Suppose the set R of alternative outcomes of an action is finite. It is possible to label the outcomes $r_1, \ldots, r_N$ in R in such a way that, for all r_i in R,

(6.25) $$r_1 \underset{R}{\prec} r_i \underset{R}{\prec} r_N.$$

To avoid an uninteresting degenerate case, suppose that r_N is strictly preferred to r_1:

(6.26) $$r_1 \underset{R}{\prec} r_N.$$

For any action α, let $\sigma_i(\alpha)$ be the event that α has the outcome r_i:

(6.27) $$\rho(x, \alpha) = r_i \qquad \text{for } x \text{ in } \sigma_i(\alpha).$$

For each action α, the events $\sigma_1(\alpha)$, ..., $\sigma_N(\alpha)$ form a partition of X. Considering another outcome r_j in R, $j \neq i$, the following theorem can be obtained about υ on R.

Theorem 3 (The Property of Utility Function υ): A utility function υ on R, which enables the expected utility function $\Omega(\alpha)$ on action α to represent the preference order of actions, if and only if

$$\upsilon(r_i) - \upsilon(r_j) > 0. \text{ Where } r_i \underset{R}{\prec} r_j.$$

To prove the theorem, a specific case in which the set R consists of only two outcomes is used, doing this, first because the case of two outcomes is simple and easy to understand, and second a complete proof is no different to the simple case except in using "bench-marking" and successive approximation which have already been described in the last section.

Proof: Let the two outcomes in R be denoted by s (success) and f (failure), following (6.25) and (6.26)

(6.28) $$f \underset{R}{\prec} s.$$

For any action α, let $\sigma(\alpha)$ denote the set of states for which α results in success, that is a special case of (6.27),

(6.29) $$\rho(x, \alpha) = \begin{cases} s & \text{for } x \in \sigma(\alpha) \\ f & \text{for } x \notin \sigma(\alpha). \end{cases}$$

From the definition of $\Omega(\alpha)$ (6.7), and (6.17),

(6.30)
$$\Omega(\alpha) = \sum_{i=1}^{N} \upsilon(r_i)\, \pi[\sigma_i(\alpha)] = \upsilon(s)\pi(\sigma[\alpha]) + \upsilon(f)[1 - \pi(\sigma[\alpha])]$$
$$= [\upsilon(s) - \upsilon(f)]\pi[\sigma(\alpha)] + \upsilon(f).$$

This means if $\upsilon(s) - \upsilon(f) > 0$, the ordering of actions defined by $\Omega(\alpha)$ will be the same defined by subjective probability function π on event $\sigma(\alpha)$. Notice that if

(6.31)
$$\upsilon(s) = 1,$$
$$\upsilon(f) = 0.$$

The value of expected utility (6.30) of an action is equal to the probability of success.

In two outcome actions, for example α and α', from the definition of the probability ordering $\underset{X}{\prec}$, (6.13) and (6.14), the ordering of the actions is the same as the ordering of the events;

$$\alpha \underset{A}{\prec} \alpha' \qquad \text{if and only if} \qquad \sigma \underset{X}{\prec} \sigma',$$

where σ is the event, which when it happens action α will result in success, otherwise failure; similarly, event σ' is the event, which when it happens action α' will have the result of success, otherwise failure.

In addition, according to the definition of subjective probability (6.16), that

$$\sigma \underset{X}{\prec} \sigma' \qquad \text{if and only if} \qquad \pi(\sigma) \leq \pi(\sigma'),$$

Therefore,

$$\alpha \underset{A}{\prec} \alpha' \qquad \text{if and only if} \qquad \pi(\sigma) \leq \pi(\sigma').$$

This is the two actions α and α', each can result in either success or failure, then the preferred action is the one with the highest probability of success. In other words, it can be said that for actions with two outcomes, the preference order of actions is the same as ordering by probability of success.

Hence the preference ordering $\underset{A}{\prec}$ can be represented by the expected utility function Ω if and only if

(6.32) $$\upsilon(s) - \upsilon(f) > 0.$$

End of Proof.

2. Measurement utility function υ

Suppose, again the set R is finite. The outcomes $r_1, ..., r_N$ in R can be labelled in a way that, for all r_i in R, $r_1 \underset{R}{\prec} r_i \underset{R}{\prec} r_N$ (6.25), and $r_1 \underset{R}{\prec} r_N$ (6.26).

For any action α, let $\sigma_i(\alpha)$ be the event that α has the outcome r_i:

$$\rho(x, \alpha) = r_i \qquad \text{for } x \text{ in } \sigma_i(\alpha).$$

Notice that, for each action α, the events $\sigma_1(\alpha)$, ..., $\sigma_N(\alpha)$ form a partition of X. Following an approach suggested by the two-outcome case in the above proof, first define the utilities of the *worst* and the *best* outcomes in R to be 0 and 1, respectively,

(6.33) $$\upsilon(r_1) = 0,$$
$$\upsilon(r_N) = 1.$$

To measure the utilities of the other outcomes in R, a class of benchmark actions can be constructed. From the theorem of the measurement of an event's probability, for each number k between 0 and 1, an event W_k can be found such that $\pi(W_k) = k$. Define the action α_k by

(6.34) $$\rho(x, \alpha_k) = \begin{cases} r_N & \text{for } x \text{ in } W_k \\ r_1 & \text{for } x \text{ not in } W_k \end{cases}.$$

The expected utility of the benchmark action α_k is immediately seen to be k:

(6.35) $$\Omega(\alpha_k) = k.$$

Now any outcome r_i in R. Since

$$r_1 \underset{R}{\preceq} r_i \underset{R}{\preceq} r_N,$$

applying definition (6.34) with $k = 0$ or 1, thus,

(6.36) $$\alpha_0 \preceq r_i \preceq \alpha_1.$$

This comparison between actions and outcomes is meaningful by virtue of the Independence condition 2 and the Sure-Thing for Outcomes; hence the letters R and A under the $\preceq$ symbol can be omitted without risk of ambiguity. From (6.36), there is

some number k such that $r_i \sim \alpha_k$; we can define that number k to be the utility of r_i. To measure k, the same process of successive approximation that was used to measure the subjective probability of an event can be used. So, the utility of an outcome r_i in R is

(6.37) $$r_i \sim \alpha_{U(r_i)}.$$

6.5 Summary - Expected Payoff of an Action

In this chapter, a theory is provided that explains why an agent chooses a particular action among a set of alternatives. The theory is maintained within a clear boundary regarding to its origin, scope, and basic definitions. The agents are not allowed to act in an "awkward manner" such as in the CPS model described in Chapter 4 where agent i

has a goal φ and knows that the action α can achieve the goal φ, at the same time agent i has the goal of not performing action α without specifying why the agent does not want to perform action α. The definitions of the theory foundations in the first section stated three basic points: 1) agents are rational in the sense that the rationality coincides with human agent's rationality. 2) The development of the theory is based on a general philosophical approach rather than a particular practice. 3) The action is divisible in some way. Study is not focused on a pure action at an abstract level, instead the action is an act in the sense that the focus is on its result rather than its name.

The decision theory developed in this thesis showed that a decision is controlled by a mental state called preference that is further defined in two elemental states called tastes and beliefs. Tastes describe a mental state where an agent has a particular order of alternative actions; beliefs describes a mental state of an agent where it has a particular order of future alternative events.

To find a single continuous quantity variable which can be used to express and explain the decision-making on actions under uncertainty, expected utility was introduced. Each outcome r is a result of an action α in a given state x of the environment. Thus $r = \rho(x, \alpha)$, where ρ is the outcome function. If $\sigma_i(\alpha)$ denotes the event "action α results in outcome r_i," then the expected utility for action α is

$$\Omega(\alpha; \rho, \pi, \upsilon) = \frac{1}{N} \sum_{i=1}^{N} \upsilon(r_i)\, \pi[\sigma_i(\alpha)], \tag{6.38}$$

where π is the *subjective probability function* and υ is the *utility function*. The expression on the left-hand side of (6.38) emphasises that the expected utility depends on the decision-maker's action only, given the functions ρ, π *and* υ. These functions summarise the factors beyond an agent's control: its beliefs π, its tastes υ, and its idea of

the "physical" relation ρ that states how outcomes are determined by itself and by the environment. An agent's action α, on the other hand, is under its control. Therefore, rational agents choose an action with the greatest *expected utility*.

To demonstrate the possibility of quantifying the expected utility, the independence between an agent's tastes and beliefs were explored and the quantisation of the two "auxiliary" numerical function were also demonstrated.

To provide a better explanation of the theory, the expected utility function can be simplified.

Since $r = \rho(x, \alpha)$, and the utility of r is $\upsilon(r)$, the utility of an outcome can be directly expressed as a function of x and α:

(6.39) $$\upsilon(r) = \upsilon\,[\rho(x, \alpha)] = \omega(x,\alpha).$$

Here ω is called the payoff function; it is equivalent to the successive application of the outcome function and the utility function. It is thus a combined expression of an agent's tastes and of its explanation of the outcome as determined by its action and the environment.

Suppose that the set X of alternative states of the environment is finite, and write $\phi(x)$ for the probability of the state[8], x, $\phi(x) = \pi(\{x\})$. The function ϕ is usually called a *probability density function* in Mathematics. The expected utility of an action can now be written in a simpler form[9]

(6.40) $$\Omega(\alpha;\, \omega,\, \phi) = E\omega(x,\alpha) = \frac{1}{N}\sum_{i=1}^{N} \omega(x,\, \alpha)\, \phi(x) = \frac{1}{N}\sum_{i=1}^{N} \upsilon[\rho(x,\, \alpha)]\, \phi(x).$$

[8] Strictly speaking, we should say "probability of the set consisting of the single element x," which we have denoted by {x}.

[9] Where Eω is a notion of function called expected payoff.

It is called the *expected payoff* of the action α. Finally, we shall say that a rational agent's behaviour under uncertainty agrees with the expected utility principle. That is,

> **for a rational agent, there exists a probability distribution ϕ on the set of the states of environment, an outcome function ρ by which it explains the outcome as a joint result of its and nature's actions, and a utility function υ on the set of outcomes, its preference between actions depends on the expected payoff of the actions. The agent always prefer the action with higher expected payoff.**

Formally,

(6.41) $$(\text{Prefer } i\ \alpha\ \alpha') \stackrel{def}{=} E\omega(\alpha) \geq E\omega(\alpha').$$

Chapter 7

Formalisation of the SMM Model

In one way or another, we are forced to deal with complexities, with "wholes" or "systems" in all fields of knowledge. This implies a basic re-orientation in scientific thinking.

-- Ludwig von Bertalanffy

This chapter, as the last part of theoretical contributions of the thesis, will provide a formal model of Shifting Matrix Management [Li et al. 97]. It is hoped that the formalisation of the SMM framework can provide some basis for a further mathematical framework and development of proofs and can cover a wider range of applications by serving as an abstract, top-level specification for building multi-agent systems.

There are three primary objectives for developing a formal SMM model:

- To provide a complete, comprehensive, rational formulation of multi-agent cooperation in a complex, dynamic and open environment.
- To produce a theory which is mathematically tractable.

- To offer a clear mapping from the theory to its implementation.

These criteria mean that the SMM model must not only provide a theory that can specify any agent's behaviour from its internal and mental perspectives (e.g., motives, beliefs, and goals), but also provide a procedural guideline for implementation in which its applicability can be clearly illustrated.

This chapter is arranged in the following manner. Section 7.1 provides an introduction of a collection of logical preliminaries to pave the way for the formal presentation because the logical preliminaries define the formal language used in the description. In section 7.2, models of events and actions are developed. The agents' mental states are viewed as events under the *possible worlds* semantics, and the agents' actions are viewed as particular types of events that occurred because of the agents' performance. With these views, the agents' attitudes and behaviour used in the formal SMM model are then defined and formalised. In section 7.3, a twin-notion of *joint commitments* and *social conventions* are defined. Upon this a notion of *joint intention* is defined for agents' goal justification in a social context. In section 7.4, the SMM framework proposed in Chapter 5 is formalised as an abstract model of multi-agent systems that can be used for both theoretical proofing and practical applications. Finally, section 7.5 provides a summary and the distinctive features of the formal SMM model are identified.

7.1 Logical Preliminaries

The *possible worlds* semantics, temporal logic, dynamic logic and a collective of derived operators are introduced in this section.

7.1.1 Possible Worlds

The "classical" model for reasoning about knowledge and belief is the so-called *possible worlds* model [Reeves and Clarke 90]. The basic idea is that besides the actual state of affairs there are a number of other states (or worlds). *Possible worlds* therefore represent not only actually occurring worlds but also any logically possible situations in which there are no contradictions.

Possibility, which is not the same as conceivability, is a more general notion. For example the statement "London buses are black" is true in some imaginable world, meaning it is possible. The possibility is not ruled out just because it is not actually the case. However a proposition such as "the sky is blue and the sky is not blue" cannot be true in any possible world since it is contradictory [Reeves and Clarke 90]. Formulae formed by using this model are no longer true or false absolutely; rather they are true or false with respect to a particular world. For example, it is possible to state the expression "robot R_o can cooperate" is true in world w_0 but false in some other imaginable world such as world w_{12} where "robot R_o can not cooperate." Another key idea is the notion of accessibility, indicating which worlds are accessible from which others. In possible world semantics, the statement "robots can cooperate" has the value true with respect to a particular world w_i if and only if this statement is true in all worlds accessible from w_i. The statement "robots can cooperate" is false with respect to a particular world w_j means there must be some possible worlds which are accessible from the world w_j in which the statement is true and some in which it is false.

7.1.2 Temporal Logic

One important use of *possible worlds* semantics is to relate the truth values of a proposition to time. In this case, the accessibility relation can be viewed as connecting

states or possible worlds that are temporally earlier than those which are temporally later.

In temporal logic, the model is a triple ($\otimes$, X, $\mathfrak{R}$) containing a data domain $\otimes$, a set of states X and an interpretation $\mathfrak{R}$ giving meaning to every predicate symbol. Where, $\otimes$ has only two integers *true* and *false*. $\mathfrak{R}$ represents the accessibility relation. A proposition x in X, will have an interpretation $\mathfrak{R}[x]$ which maps x onto truth value, that is,

$$\mathfrak{R}[x] \in (\otimes \rightarrow \{true, false\}).$$

To emphasise the fact that $\mathfrak{R}$ represents an "earlier-later than" relation it will be written using "<". The fact that a proposition x in X is true at a particular time t is represented as

$\mathfrak{R}_t[x] = true$. This is denoted as: $t \models x$.

A proposition x that is true in all possible worlds is written $\models x$. The four modalities that are used in the following descriptions are listed below:

Table 7.1 Temporal Logic Primitives

PRIMITIVE	MEANING
Ox	x is true in the next state. i.e. $t \models Ox$ if and only if $t+1 \models x$.
$\Diamond x$	there is a future time at which x will be true (eventually x). i. e. $t \models \Diamond x$ if and only if there is a t' such that $t < t'$ and $t' \models x$.
$\Box x$	at all future times x will be true (always x). i.e. $t \models \Box x$ if and only if for all t', $t' \models x$.
$x \mu x'$	x' will be true in some future state, and x will be true (at least) until then.

7.1.3 Dynamic Logic

The *possible worlds* semantics can also be used to express relations between actions within a computation providing a form of dynamic logic. This is achieved if the possible worlds are interpreted as states of a computation and the accessibility relation links one state of the computation to another. In this case, accessibility is dependent on the program being executed, thus the accessibility relation needs to be indexed by the program. If σ and σ' are states of a computation (in an environment state $x \in X$, σ can be regarded as an *event* that the computation take places in x) and we get from σ to σ' by executing program α then $(\sigma, \sigma') \in \mathfrak{R}(\alpha)$. Therefore, it is possible to define the following primitives that can be used to express the relations between actions in a program.

Table 7.2 Dynamic Logic Primitives

PRIMITIVE	MEANING
$\alpha ; \alpha'$	do action α then do action α'.
$\alpha \mid \alpha'$	do action either α or α'.
α^*	do action α iteratively.
$\alpha?$	evaluate the proposition about action α. If it is true then proceed with the evaluation, otherwise fail.

7.2 Event and Action Models

With the logical primitives described in the previous section, it is possible to develop higher models of an agent's behaviour. The most comprehensive formalisation is developed by Cohen and Levesque [Cohen and Levesque 90(1)(2)]. In order to formalise an individual agent's intention, they have provided a theory of event and action. Cohen and Levesque's formalisation is described in a model language which has the usual connectives of a first-order language with operators for prepositional attitudes

and for talking about sequences of events and actions [Cohen and Levesque 90(1)]. This thesis adopts their language and only descriptive semantics are given. The detailed syntax and semantics of the language and its proofs can be found in [Cohen and Levesque 90(1)]. Appendix C lists the syntax of the language used in this thesis.

7.2.1 Event and Action Primitives

In the event model, possible worlds are temporally extended into the past and future. Each such world consists of an infinite sequence of primitive events. Each event is of a certain type (a proposition denoted by *p*) and can have an agent associated with it[1]. For example, beliefs are formalised as accessibility relations over possible worlds, where the accessible worlds are those which have been selected as most desirable and the beliefs are true in all these worlds. The primitives of events and other events examples are listed in Table 7.3.

Table 7.3 Events Primitives and Examples

OPERATORS	MEANING
(Happened *e*)	event *e* has just happened.
(Happening *e*)	event *e* is happening now.
(Happens *e*)	event *e* happens next.
(Agts *e* i_1, i_2,..., i_n)	i_1, i_2,..., i_n are precisely the agents required to the event *e*.
(Bel *i p*)	agent *i* has a belief *p*.
(Mot *i p*)	agent *i* has a motive *p*.
(Goal *i p*)	agent *i* has a goal *p*.
(Know *i p*)	agent *i* has a knowledge of *p*.

[1]Symbol "*e*" is used to denote an event that has an agent associated with it. It is used only to distinguish from the normal events denoted by "σ" in the *possible worlds* semantics.

Action is viewed as a particular type of event that happened because of an agent's performance. Action expressions are built up using both event primitives and dynamic logic primitives. For example, (α; α') means the action α is immediately followed by action α' and (Happens α) means that action α happens next. The primitives of action used in this thesis are listed in Table 7.4.

Table 7.4 Actions Primitives

OPERATORS	MEANING
α ; α'	action α immediately followed by action α'.
α | α'	either action α or α' happened next.
α^*	action α iterated.
Happened α	action α has just happened.
Happening α	action α is happening now.
Happens α	action α happens next.
Agts(α i_1, i_2,..., i_n)	i_1, i_2,..., i_n are precisely the agents required to perform the action α..

To explicitly express and emphasise the agent that performs the action, a set of agent's actions is defined. An agent i's action is described by the action primitives associated with the agent. (Done α i), (Doing α i) and (Does α i) means agent i has done, is doing and does action α next. (Doesn't α i) means agent i is not doing action α next in any alternative future. That is,

$$\begin{aligned}
&(\text{Done } \alpha\, i) \stackrel{def}{=} (\text{Happened } \alpha) \wedge (\text{Agts } \alpha\, i),\\
&(\text{Doing } \alpha\, i) \stackrel{def}{=} (\text{Happening } \alpha) \wedge (\text{Agts } \alpha\, i),\\
&(\text{Does } \alpha\, i) \stackrel{def}{=} (\text{Happens } \alpha) \wedge (\text{Agts } \alpha\, i),\\
&(\text{Doesn't } \alpha\, i) \stackrel{def}{=} \neg(\text{Happens } \alpha) \wedge (\text{Agts } \alpha\, i).
\end{aligned} \tag{7.1}$$

7.2.2 Agent's attitudes

In the above event model, an agent's mental states or attitudes at a given time are regarded as events in the possible worlds. The two fundamental points in the model are:

1. The attitudes of an agent characterises what *the world would be like* if the agent's mental states are true, rather than what the agent's actively and explicitly mental states are. So it is an *implicit* expression of the agent's attitudes[2].

2. No agent's mental states changes without some event happening. In particular, there is no notion in this model for the simple passage of time (without any intervening events) affecting any agent's mental states. This event model provides a base where agent's attitudes and their relationships can be studied and analysed with logic models.

However, there are disparities in multi-agent systems research, as in philosophy, about an agent's basic attitudes and the relationships between these attitudes. For example, Cohen and Levesque's model uses two basic attitudes: goal and belief and other attitudes are defined in terms of these two basic attitudes [Cohen and Levesque 90(1), (2)]. In contrast, Rao and Georgeff's model has three basic attitudes: belief, desire, and intention [Rao and Georgeff 91, 93]. This thesis, as stated in Chapter 5 (page 117), is not concerned with the debate about complex philosophical concepts and the relationships between concepts such as motive, goal, desire, belief and so forth. Instead, the thesis is trying to provide a self-contained theory that can 1) explore aspects, as much as possible, associated with multi-agent cooperation; 2) produce a computational tractable theory; and 3) offer a clear mapping from the theory to its implementation.

[2] The issues involved in an agent's explicit versus implicit attitudes are explored in [Levesque 84].

In the SMM model, the attitudes of an agent, such as wants, desires, and hopes are not included because these attitudes need not be consistent. Typically, one agent can desire more gain and less work at the same time (we assume that gain can only be obtained from work) just like humans can desire having ten chocolate bars a day and losing weight at the same time (assume that ten chocolate bars can put on weight significantly). The SMM model, similar to other intention models in multi-agent systems [Cohen and Levesque 90(1), (2)], assumes that once an agent has sorted out its possibly inconsistent desires in deciding what to achieve, the world that agents are situated in is consistent.

According to the previous experience in building multi-agent systems, this suggests that any systems would have their agents possess consistent attitudes. In addition, one particular application can always find the most explicit and convenient representation of an agent's attitudes. It is difficult in an application to have its agents possess implicit mental states like beliefs, goals in the system set up. Another problem remains in choosing and representing an agent's initial attitudes against the agent's autonomy. The more detailed attitudes an agent has the less autonomy it possesses. For example, if an agent initially has a goal of performing a task, the agent would not have any autonomy on the task selection. The agent will not reason about different tasks: the benefit and the cost of performing the tasks. Consequently the system appears less intelligent.

1. Motive and Capability. The SMM model uses these attitudes as the basic attitudes. Other attitudes are derived and expressed by these two basic attitudes. It has the advantage of easy and explicit representation in applications. Motive represents the general objectives that an agent holds initially. Capability represents an agent's initial reasoning ability and physical functionality in a multi-agent system. Both motive and

capability are propositions which can be expressed explicitly in an appropriate syntax according to the language used in the multi-agent systems.

The motive that an agent possesses initially in a multi-agent system can be regarded as a long term goal of the agent. It governs the agent's short term goal selection and adjustment. The notion of goal in this model is subordinate to the notion of motive. For example, one can set an agent's motive to "*gain profits as much as possible.*" In an actual task's performance process, the agent will, after checking its capability to perform the tasks and evaluating each task performance in terms of possible profits, set up its short term goal - *to perform a selected task.* One could set different motives for different agents to achieve possible system efficiency. Thus, (Mot i ζ) is denoted to express an agent i has a motive ζ[3]. We have

Proposition 1. *If* $\models \zeta$, *then* $\models$ (Mot i ζ).

If ζ is a proposition that states agent i's motive set by users during the system set up, then the agent will hold this motive at all times.

An agent's capability is defined as consisting of two distinct components: knowledge and ability. An agent's knowledge is denoted by (Know i p) which represents agent i having a knowledge about proposition p. An agent's knowledge is a fundamental mental state of any other deliberative behaviours of the agent. In this thesis I propose that because agents are made artificially, we should, and could, prevent problems that have occurred in human society from appearing in the agents' world. In other words agents should not reason or believe something or anything about which they do not have any

[3] Letter "ζ" is used here purely for distinguishing motives from other propositions. In fact, motive is still a proposition.

knowledge in any form. It may sound strong in theory but it does have benefits in application. Similarly,

Proposition 2. *If* $\models p$, *then* $\models$ (know i p).

That is, if p is a proposition that states agent i's knowledge in the system set up, then agent i will hold this knowledge in all its chosen worlds.

An agent's ability denoted by (Can i α) represents that the agent i has the physical capability of performing action(s) α.

Proposition 3. *If* $\models \alpha$, *then* $\models$ (Can i α).

That is, if α is an action(s) that is attached with an agent i in the system set up by the user, then agent i will hold this ability in all the possible world at all times.

2. Goal and Belief. Theoretically, we could adopt Cohen and Levesque's goal and belief models. They are accessible relations from the current world to other possible worlds. However, to bridge the gap between theory to application and to provide clear guidelines for applications, this thesis defines goal and belief as results of three basic attitudes: motive, knowledge and ability after certain events have happened.

An agent i having a goal φ denoted by (Goal i φ) is an event[4]. It happens when agent i perceives the tasks to be performed in a system, which by having the goal of performing the tasks the agent will be a step closer to fulfilling its motive ζ. In addition, an agent is not allowed to have a goal that it has no knowledge about, and cannot contribute anything to the goal achievement. In formal language,

[4] Letter "φ" used here is purely for distinguish goal with other propositions. In fact, goal is still a proposition.

(7.2) $(\text{Goal } i\ \varphi) \stackrel{def}{=} (\text{Mot } i\ \zeta) \wedge (\text{know } i\ [\exists\ (\varphi)\ (\varphi\ \mu\ \zeta) \vee (\varphi\ \mu \Diamond\ \zeta)]) \wedge$

$$\left[\begin{array}{l}(\text{Know } i\ [\exists\ (\alpha)\ (\text{Can } i\ \alpha) \wedge ((\text{Done } \alpha\ i);\ \varphi) \vee \\ (\text{Know } i\ [\exists\ (\alpha, \alpha')\ (\alpha \subset \alpha') \wedge (\text{Can } i\ \alpha) \wedge ((\text{Done } \alpha\ i);\ \Diamond\varphi)\end{array}\right].$$

The square brackets used in the above formula and later in the thesis mean that they contain complex events. That is, an agent i has a goal φ, which means that relative to its motive ζ, it knows there is an existing goal φ. When φ is satisfied ζ will be satisfied or after φ becomes satisfied, ζ will be eventually satisfied. In addition, agent i knows that: 1) there are some possible complex actions α of which agent i has the ability to perform them alone and after it has performed α, φ will be satisfied; 2) there are some complex actions α' of which agent i can only perform part, that is action α, after i performs α, φ will be eventually satisfied. In this case we say that the agent i is *dependent* on other agents performing certain actions (α'-α) to bring about its goal φ. In order to make the theory complete, the definition of the dependence relationship is provided. A social dependence, denoted by (S-Dep i i' α φ), means agent i is dependent on agent i' to achieve goal φ regarding action α. It has,

(7.3) $$(\text{S-Dep } i\ i'\ \alpha\ \varphi) \stackrel{def}{=} (\text{Goal } i\ \varphi) \wedge \neg(\text{Can } i\ \alpha) \wedge (\text{Can } i'\ \alpha) \wedge ((\text{Done } \alpha\ i')\ \mu\ \Diamond\varphi).$$

Similar to the goal definition, an agent i has a belief about a proposition p denoted by (Bel i p) is an event when the agent needs access to possible worlds from the current world. It is based on the agent's knowledge about such worlds and any available theorems. These available theorems refer to the true proposition in all possible worlds that are either learnt by the agent from other agents or from its own experience. Clearly, it is a strong definition that excludes an agent's unrealistic and irrational beliefs. That is,

(7.4) $$(\text{Bel } i\ p) \stackrel{def}{=} (p \wedge (\text{Know } i\ \text{p})) \vee$$
$$\exists q\ (q \wedge (\text{Know } i\ q) \wedge (p \subset q)) \vee$$
$$\exists p\ ((\text{Agts } p\ i) \wedge (\text{Happened } p)).$$

The last part of the definition states that an agent's belief with respect to some proposition p is adopted only for those propositions for which the agent has a good evidence. This evidence comes from the agent's own performance experience.

7.2.3 Other Derived Operators

A number of derived operators will be introduced in order for the SMM model to be described. Before introducing these derived operators two *path quantifiers* have to be defined. The path quantifier $\mathbb{A}$ is a *path formula*. $\mathbb{A}\varphi$ means that φ is satisfied in all the futures that could arise from the current state. The path quantifier $\mathbb{E}$ is the dual of $\mathbb{A}$. $\mathbb{E}\varphi$ means that φ is satisfied in at least one possible future state. The distinction between *path* and *state* formulae is that the state formulae are evaluated with respect to the 'current state' of the world, whereas the path formulae are evaluated with respect to a course of events. The following derived operators are based on an extension of the first-order predicates of AND ($\wedge$), OR ($\vee$), NOT ($\neg$) and conditional ($\Rightarrow$) with all primitives described above.

1. Goal Sequence. $\varphi\ \mu\ \psi$ means φ is satisfied until ψ becomes satisfied. i.e.

(7.5) $$\varphi\ \mu\ \psi \stackrel{def}{=} (\text{Happens } (\neg\psi?; \varphi?)^*; \psi?).$$

2. Achievement. (Achieves $\alpha\ \varphi$) represents an action α achieving a goal φ.

(7.6) $$(\text{Achieves } \alpha\ \varphi) \stackrel{def}{=} \mathbb{A}((\text{Happens } \alpha) \Rightarrow (\text{Happens } \alpha; \varphi?)).$$

3. Persistent Goal and Intention. The persistent goal and intention were originally developed by Cohen and Levesque [Cohen and Levesque 90(2)] to explain agents' social behaviour. It is used in this thesis to express the mental states order of an agent during a cooperation process. The *persistent goal* is adopted to capture a mental state of the agent that has committed itself to achieving a goal relative to a motive, which characterises the justification for agent's commitment. An agent i commits itself to achieving goal φ relative to motive ζ is denoted by (P-Goal i φ ζ). It has been defined as: (i) agent i believes that φ is currently false, (ii) that agent i wants φ to be eventually true and (iii) that this state of affairs will continue until i comes to believe either that φ is true or that it will never be true or that ζ is false.

$$(7.7) \qquad (\text{P-Goal } i\ \varphi\ \zeta) \stackrel{def}{=} ((\text{Bel } i\ \neg\varphi) \wedge (\text{Goal } i\ \Diamond\varphi)) \wedge \mathbb{A}((\text{Goal } i\ \Diamond\varphi)\ \mu\ [(\text{Bel } i\ \varphi) \vee (\text{Bel } i\ \neg\Box\varphi) \vee (\text{Bel } i\ \neg\zeta)]).$$

The above persistent goal definition clearly distinguishes it from a normal goal. That is the agent commit itself to achieve the goal and hold this commitment until it has been achieved or another event has been perceived. The reason for an agent having a persistent goal may be because 1) the other goals may not be currently achievable or 2) among all the currently achievable goals, goal φ is the most favourable goal to fulfil its motive ζ.

An *intention* is then defined as an agent has a persistent goal of achieving φ. (Intend i φ ζ) is denoted to represent that agent i *intends* to bring about goal φ relative to a motive ζ. It has been expressed in the following manner:

(7.8) (Intend i φ ζ) $\stackrel{def}{=}$ (P-Goal i

$\exists e$(Done [(Bel i $\exists e'$(Happens e'; φ?)) $\wedge$

$\neg$(Goal i $\neg$(Happens e;φ?))]?;e;φ?) ζ).

The above definition does not only state that an agent's intention about a goal φ has to be its persistent goal, but also states that 1) to bring about φ, the agent is committed to doing some sequence of events e, after which φ holds. 2) To avoid intending to make φ true by committing himself to doing something accidentally or unknowingly, the agent needs to think it is about to do *something* (event sequence e') to bringing about φ. Generally, an agent can believe this if the agent in fact has a plan for bring it about. However, it is quite difficult to define what a plan is, or define what it means for an agent to have a plan [Pollack 90]. A way, not too far off though, is to require that an agent believes it is about to do something (event sequence e' here) that will bring about φ. Finally 3) the intention definition requires that prior to doing e to bring about φ, the agent does not have as a goal that e is not bringing about φ.

4. Prefer and Decide. Prefer and decide are two important mental states of an agent in dealing with alternative actions. They have been hitherto neglected by agent theorists. One reason is because the research in developing agent theory is mainly focused on the attitudes and relations that are needed to support finding the necessary conditions of an agent's basic, non-social behaviour. The other reason is that both attitudes are complex and difficult to be formalised. Firstly, both attitudes involve other aspects such as the perspectives that an agent considers in the time being and the standard that the agent uses to evaluate the alternative actions. Secondly, it is extremely difficult to predict the chance of an expected event happening in dynamic worlds. Finally, it is equally difficult to reason about other agents' mental states and beliefs. Nevertheless, Chapter 6 of this

thesis provides a comprehensive theory about the factors that a decision maker considers and a possible way that these factors can be quantified. It is summarised here. (Prefer i α φ) is denoted to express an agent i who prefers action α in achieving a goal φ. We have,

(7.9) $\quad$ (Prefer i α φ) $\stackrel{def}{=}$ (Goal i φ) $\wedge$

(Bel i $\exists\alpha \exists\alpha'$((Achieves α φ) $\wedge$ (Achieves α' φ) $\wedge$ [$E\omega(\alpha) \geq E\omega(\alpha')$])).

That is, an agent i who prefers action α in achieving a goal φ is a mental state of the agent. It happens if the agent has a goal φ, and the agent believes that there are existing actions α and other actions α'. These actions all could bring about the goal φ. But the expected payoff ($E\omega$) from action α is greater than the expected payoff from other actions α'. With this definition the decision is easy to formalise. (Decide i α φ) is used to express an agent i who decides to adopt action α to bring about a goal φ. We have,

(7.10) $\quad$ (Decide i α φ) $\stackrel{def}{=}$ (Intends i φ ζ) $\wedge$ (Prefers i α φ).

That is, the mental state 'decide' of an agent is only held if the agent has an intention of bring about the goal φ relative to its initial motive ζ and the agent prefers action α to any other actions.

5. Attempts. The model of attempts is also developed by Cohen and Levesque [Cohen and Levesque 90(2)] to express an agent's effort towards a state of the world. In their definition an attempt by an agent i to bring about a state φ is an action α performed by i with the goal that after α is performed, φ is satisfied, or at least ψ is satisfied.

The ultimate goal of the attempt — the thing that i hopes to bring about — is represented by φ, whereas ψ represents "what it takes to make an honest effort". Bringing about ψ will be sufficient, and eventually cause φ. That is:

$$(7.11) \qquad (\text{Attempts } i\ \alpha\ \varphi\ \psi) \stackrel{def}{=} \left[\begin{array}{l}(\text{Bel } i\ \neg\varphi)\ \wedge\ (\text{Agt } \alpha\ i)\ \wedge \\ (\text{Goal } i\ (\text{Achieves } \alpha\ \varphi))\ \wedge \\ (\text{Intend } i\ (\text{Does } (\alpha\ \psi?)i)\ \zeta)\end{array}\right]?;\alpha\,.$$

The above definition captures the intuitional characteristic of an attempt. That is an attempt may result in success or failure. However, it does not state the conditions of failure. A rational agent should know to what extent it should drop its attempt, and try to achieve ψ rather than φ. A modified attempt explicitly states these conditions. It is based on the decision theory of actions. Because an attempt is an action, which will consume resources of the performing agent, the agent will evaluate the attempt against its expected payoff. The attempt is now summarised in the following formula.

$$(7.12) \qquad (\text{Attempts } i\ \alpha\ \varphi\ \psi) \stackrel{def}{=} [(\text{Decides } i\ \alpha\ \varphi)?;(\text{Does } \alpha\ i)]\ \mu$$
$$[(\text{Bel } i\ (\text{E}\omega(\text{Doing } \alpha\ i) < \text{E}\omega(\alpha)))?;\ (\text{Intends } i\ \psi\ \zeta)].$$

That is, an agent i's attempts to bring about φ relative to its motive ζ is an action. It is performed by the agent i if it decides to adopt action α to bring about φ, then the agent does action α until it believes that the expected payoff of doing action α is less than the expected payoff of action α (because of changes in the world). The agent then will intends to bring about ψ rather than φ. However, bringing about ψ will be sufficient, and eventually will cause φ.

7.2.4 Collective and Group Related Events and Actions

In a multi-agent situation, it is necessary to express collective events. The letter g is used to denote a group of agents. Therefore (Agts α g) means that the group g are precisely the agents required to perform an action or action sequence α. (Singleton g i) means g is a singleton group with i as the only member.

(7.13) $(\text{Agts } \alpha\, g) \stackrel{def}{=} (\forall i(\text{Agts } \alpha\, i)) \Rightarrow (i \in g)$.

(7.14) $(\text{Singleton } g\, i) \stackrel{def}{=} \forall i'((i' \in g) \Rightarrow (i' = i))$.

(7.15) $(\text{Agts } \alpha\, i) \equiv \forall i((i \in g) \wedge (\text{Agts } \alpha\, g)) \Rightarrow (\text{Singleton } g\, i)$.

A number of derived events are defined to express the mental states of a group of agents. (M-Bel g φ), (M-Goal g φ), (M-Prefer) and (J-Decide) are used to represent the mutual belief, mutual goal, mutual preference and joint decision in a group of agents g respectively. (M-Know g φ) and (J-Can g φ) are used to express the mutual capability of a group of agents g. (M-Dep g φ) is used to express all the agents in a group g mutually depend on each other to achieve their mutual goal φ. Finally, a common goal φ in an agents group (C-Goal g φ) is a mutual goal in the group such that agents mutually depending on each other to achieve it. They are formalised as follows:

(7.16) $(\text{M-Bel } i, i', \text{p}) \stackrel{def}{=} \text{Bel}(i, \varphi \wedge \text{M-Bel}(i', i, p))$.

(7.17)[5] $(\text{M-Bel } g\ \varphi) \stackrel{def}{=} \forall i\ ((i \in g) \Rightarrow (\text{Bel } i\ (\varphi \wedge (\text{M-Bel } g\ \varphi))))$.

(7.18) $(\text{M-Goal } g\ \varphi) \stackrel{def}{=} \forall i\ ((i \in g) \Rightarrow (\text{M-Bel } g\ (\text{Goal } i\ \varphi)))$.

(7.19) $(\text{M-Prefer } g\ \alpha\ \varphi) \stackrel{def}{=} \forall i\ ((i \in g) \Rightarrow (\text{M-Bel } g\ (\text{Prefer } i\ \alpha\ \varphi)))$.

(7.20) $(\text{J-Decide } g\ \alpha\ \varphi) \stackrel{def}{=} \forall i\ ((i \in g) \Rightarrow (\text{M-Bel } g\ (\text{Decide } i\ \alpha\ \varphi)))$.

(7.21) $(\text{M-Know } g\ \varphi) \stackrel{def}{=} \varphi \wedge (\text{M-Bel } g\ (\text{M-Bel } g\ (\text{M-Know } g\ \varphi))$.

(7.22) $(\text{J-Can } g\ \varphi) \stackrel{def}{=} \exists\alpha \exists g'((\text{M-Know } g\ (g' \subseteq g) \wedge (\text{Agts } \alpha\, g') \wedge (\text{Achieves } \alpha\ \varphi)) \vee$
$(\text{M-Know } g\ (g' \subseteq g) \wedge (\text{Agts } \alpha\, g') \wedge (\text{Achieves } \alpha\ (\text{J-Can } g\ \varphi))))$.

[5] This definition captures the mutual belief defined by logicians and philosophers that there is an infinite conjunction of beliefs about other agents' beliefs about other agents' beliefs and so on to any depth about a goal. It is not realisable in systems that admit the possibility of failed communication. However, it is a valuable abstraction tool for understanding multi-agent systems as it represents an ideal mutual mental state of a group of agents. It can be realised with certain practical assumptions.

(4.23) $(\text{M-Dep } g\ \varphi) \stackrel{def}{=} (\text{M-Goal } g\ \varphi) \wedge \forall i\ ((i \in g) \Rightarrow \exists \alpha\ (\text{S-Dep } i\ g\ \alpha\ \varphi))$.

(7.24) $(\text{C-Goal } g\ \varphi) \stackrel{def}{=} \forall i\ ((i \in g) \Rightarrow (\text{M-Bel } g\ (\text{M-Dep } g\ \varphi)))$.

7.3 Joint Commitment, Social Convention, and Joint Intention

Intention definition in an individual agent embodies the critical notions of *commitment* and its underlying *convention*. Commitment determines the degree to which an agent persists to achieve an intention and convention determines the rationality of an agent's behaviour when reconsidering an intention, that is the enumerated conditions under which commitment to an objective can be dropped.

Commitment and convention to an intention are even more important during social interaction where agents' actions have to be coordinated. Commitment and convention provide a base of trust upon which coordination can be applied. This is because agents can believe and expect other agents to act in a predictable way to achieve their intentions.

It is worthwhile to separate commitment and convention from intention. This is because the separation makes it possible to clearly distinguish the two concepts and to study the relationship between social commitment and social convention with joint intention.

A *commitment* was defined as a pledge or a promise; a *convention* is a means of monitoring a commitment that specifies the conditions under which a commitment might be abandoned, and how agents should behave when such a circumstance arises [Jennings 93]. In a social context where a group of agents are engaged in a cooperative activity, the agents have a social commitment to the overall aim, as well as individual commitments to the specific tasks that they have selected. This *joint commitment* is now parameterised by a *social convention*, which identifies the conditions under which the

joint commitment can be dropped, and also describes how the agents should behave. For example, if an agent drops its joint commitment because it believes that the goal will never be attained, then it is part of the notion of "cooperativeness" inherent in joint action that it informs fellow team members of its change of state.

Formally, according to [Cohen and Levesque 90(2)], a social convention denoted by (S-Convention ρ γ), is defined as a set of rules, each rule consisting of a re-evaluation condition ρ and a goal γ: if ever an agent believes ρ to be true, then it must adopt γ as a goal, and keep this goal until the joint commitment becomes redundant[6]. That is,

$$(7.25) \qquad (\text{S-Convention } \rho\ \gamma) \stackrel{def}{=} \{(\rho_k, \gamma_k) \mid k \in \{1, ..., l\}\}$$

Besides social convention, there are other parameters which are associated with joint commitments. Firstly, a joint commitment is held by a *group* of agents g. Secondly, joint commitments are held with respect to a *mutual goal*[7], φ: the goal φ is the state of affairs that the group is committed to bring about. Thirdly, joint commitments are held relative to a set of *motives* ζ, which characterises the group's goal justification. Finally, they also have a *pre-condition*, which describes what must initially be true of the world in order for the commitment to be held.

Joint commitments denoted by (J-Commit g φ ζ *Pre S-C*) have been defined as: A group

[6] The state where the commitment becomes redundant is defined as the state where the goal γ is satisfied in the original definition. It is not clear whether goal γ includes goal φ or not. However this thesis believes that the termination condition of the joint commitment should include satisfaction of the joint goal φ. It is revealed in the definition of social convention (See formula 7.28 as an example), where the goal γ implicitly states that if γ is satisfied φ will be eventually satisfied.

[7] Here, a mutual goal rather than a common goal is used to specify the precondition of joint intention. A common goal is more restricting than a mutual goal because it includes the notion of mutual dependence. However, it excludes the case where although each agent can work in isolation to achieve the goal φ, they believe working together can achieve φ more efficiently. It is the most benefit that agents expect from cooperation. Nevertheless, a mutual goal may not necessarily result in a commitment to the group before the state where joint intention is held. Joint intention may also necessarily result in achieving φ immediately afterwards. The social convention specifies the case where φ cannot be achieved immediately afterwards. In this case, agents will inform its group fellows of the latest events, which will eventually bring about φ.

g is jointly committed to a mutual goal φ with respect to each individual agent's motive ζ, pre-condition *Pre*, and social convention S-*Convention*, *S-C* in short, if: (i) pre-condition *Pre* is initially satisfied; and (ii) until the termination condition is satisfied, every agent in g either (a) has a goal of φ; or (b) believes that the re-evaluation condition of some rules ρ_k in the social convention (S-Convention ρ γ) are satisfied, and has the goals corresponding to that re-evaluation condition; where the termination condition is that the goal element of some social convention rules are satisfied (see footnote 6). Formally:

(7.26) $$(\text{J-Commit } g\ \varphi\ \zeta\ Pre\ S\text{-}C) \stackrel{def}{=} \forall i\,(i \in g) \Rightarrow Pre \wedge \mathbb{A}((P \vee Q)\ \mu\ r)$$

where, $$P \stackrel{def}{=} (\text{Goal } i\ \varphi),\ Q \stackrel{def}{=} \vee_{l=1}^{k}(\text{Bel } i\ \rho_l) \wedge \mathbb{A}[(\text{Goal } i\ \gamma_l)\ \mu\ r],\qquad r \stackrel{def}{=} \vee_{l=1}^{k}\gamma_l$$

$$Pre \stackrel{def}{=} \neg(\text{Bel } i\ \varphi) \text{ and } S\text{-}C = (\text{S-Convention } \rho\ \gamma) = \{(\rho_k, \gamma_k) \mid k \in \{1, ..., l\}\}.$$

By specifying that a social convention similar to Cohen and Levesque's joint persistent goals, joint intention is then defined in the same way as Cohen and Levesque's intention definition. Let,

(7.27) $$Pre_{JPG} \stackrel{def}{=} \forall i\,(i \in g)(\neg(\text{Bel } i\ \varphi) \wedge (\text{Bel } i\ \mathbb{E}\Diamond\varphi) \wedge (\text{Bel } i\ \text{M-Goal } g\ \varphi)),$$

(7.28) $$S\text{-}C_{JPG} \stackrel{def}{=} \left[\begin{array}{l} ((\text{Bel } i\ (\text{C-Goal } g\ \varphi)) \Rightarrow (\text{M-Bel } g\ (\text{C-Goal } g\ \varphi))) \vee \\ ((\text{Bel } i\ (\text{M-Dep } g\ \varphi)) \Rightarrow (\text{M-Bel } g\ (\text{M-Dep } g\ \varphi))) \vee \\ ((\text{Bel } i\ \mathbb{A}\Box\neg\varphi) \Rightarrow (\text{M-Bel } g\ \mathbb{A}\Box\neg\varphi)) \vee \\ ((\text{Bel } i\ \neg\zeta) \Rightarrow (\text{M-Bel } g\ \neg\zeta)) \end{array}\right].$$

The joint intention as denoted by (J-Intend g φ ζ), which is held by a group g with respect to a goal φ and motive ζ can then be defined as,

(7.29) $$(\text{J-Intend } g\ \varphi\ \zeta) \stackrel{def}{=} (\text{M-Bel } g\ (\exists e((\text{Agts } e\ g) \wedge (\text{Happens } e;\ \varphi?)) \wedge$$
$$(\text{J-Commit } g\ \mathbb{A}\Diamond(\text{Happens } e)\ \zeta\ Pre_{JPG}\ S\text{-}C_{JPG}).$$

Thus a joint intention in g to achieve φ means that agents in g mutually believe there are existing events or a sequence of events, e, which will bring about φ. The group g will jointly commit to make e happen unless one of the re-evaluation conditions specified in the $S\text{-}C_{JPG}$ is satisfied then the agents in g will drop goal φ and have the goals corresponding to that re-evaluation condition. These include, 1) when one of the agents, i, realises that the initial mutual goal of the group becomes a common goal, it will make this event known to all the group members; 2) agent i is not so sure about the initial mutual goal whether or not is a common goal, it believes that they are mutually dependant on each other to achieve goal φ, it will also make this state of affairs known to other agents; 3) agent i realises that the goal φ can never be realised or its motive ζ does not hold any more, it will make this state of affairs known to the other agents in the group. On the last occasion, to bring about φ becomes a serious problem because of changes in the environment. Joint intention ensures the state is known by all agents in the group.

With the above joint intention definition, the action of a joint attempt in a group g can be defined. A group of agents g, jointly attempts an action α, to bring about goal φ, relative to motive ζ, as denoted by (J-Attempts $g\ \alpha\ \varphi\ \psi$), is

(7.30) $$(\text{J-Attempts } g\ \alpha\ \varphi\ \psi) \stackrel{def}{=} [(\text{J-Decide } i\ \alpha\ \varphi)?;(\text{Does } \alpha\ g)]\ \mu\ [(\text{M-Bel } g\ (\text{E}\omega(\text{Doing } \alpha\ g) < \text{E}\omega(\alpha)))?;\ (\text{J-Intend } i\ \psi\ \zeta)].$$

This means group g jointly decide to adopt action α to achieve goal φ, and after α is performed, φ is satisfied, or at least ψ is satisfied.

7.4 The SMM Model

Returning to the objectives of developing a formal SMM model, that is to present a

theory that describes a cooperative problem solving process in a computationally tractable manner and to provide a clear mapping from the theory to its applications. In order to do so, the model needs to present a clear sequence of motives, beliefs, goals, intentions and actions from an individual agent to a collective of agents. The key feature of the SMM framework is its explicit expression of agents' behaviours in a cooperative process into three layers. They are the *individual layer*, the *pre-social layer* and the *social layer*. Each layer has different stages in which different behaviours for the engaged agents are required. The formal account of the SMM model is represented in the following.

7.4.1 Goal Selection

This is the beginning of a cooperation process. This stage specifies how agents choose tasks to perform. In formal terms, it states how an agent establishes its goal based on its initial attitudes. So, it is a goal selection process. As described previously, agents in the SMM model have two initial attitudes: motive and capability. The capability is further identified by two components: knowledge and ability. At the end of goal selection, the agents in the system will have settled goals. Let us consider one agent i, the pre-conditions of agent i in goal selection (PfG i φ) are:

$$(7.31)\quad (\mathrm{PfG}\ i\ \varphi) \stackrel{def}{=} (\mathrm{Mot}\ i\ \zeta) \wedge (\mathrm{Know}\ i\ \varphi) \wedge (\mathrm{Can}\ i\ \varphi) \wedge \left[\begin{array}{l} (\mathrm{Bel}\ i\ \Diamond\varphi) \wedge \\ \neg\,(\mathrm{Bel}\ i\ \neg\,(\exists\alpha\,(\mathrm{Achieves}\ \alpha\ \varphi)\ \mu\ (\mathrm{Happened}\ \alpha))) \wedge \\ (\mathrm{Bel}\ i\ [\exists\alpha\,((\mathrm{Achieves}\ \alpha\ \varphi)\ \mu\ (\mathrm{Happened}\ \alpha))]) \end{array}\right].$$

That is, with respect to agent i's motive ζ, knowledge about the possible goals, and the actions that the agent can provide towards the goal achievement, agent i selects goal φ if 1) agent i believes that goal φ is possible (it will eventually be true). 2) Agent i does

not believe that it will not bring about the goal φ (the agent does not believe that there are no existing actions, which after they have been performed, the goal still can not be achieved). And 3) agent i believes that there are actions which if they could be performed the goal will be achieved. Thus, the agent's goal selection can be expressed as,

(7.32) $$\models \forall i\,(\exists \varphi\,(\mathrm{PfC}\ i\ \varphi) \Rightarrow (\mathrm{Goal}\ i\ \varphi)),\ \varphi \text{ is a } \textit{task}.$$

That is,

> **Goal selection:** All the agents in a multi-agent system, with respect to their motive and capability, choose the goals that, they believe, are currently not realised and are achievable by certain actions being performed when some events have happened.

7.4.2 Act Selection

Act selection in the SMM model is a decision making process on alternative actions. The pre-condition for this decision making is that an agent i has a goal φ. The termination condition is that agent i decides to adopt action α to achieve goal φ. It is a complex mental behaviour of an individual agent. Firstly, this thesis agrees with Cohen and Levesque's intention theory that an agent's physical action is governed by its intention not its goal. This means an agent may have more than one goals, but at any one time the agent always has one currently intended goal. The first mental behaviour of an agent in act selection is to shift the selected goal to an intention.

Based on this more stable mental state, the next mental behaviour of the agent is to reason about the possible ways of achieving its intended goal. Among all the possible actions, the agent also needs to reason about the possibility that the expected event can

happen (by performing an action), and if the event happens, to what degree the event can bring about its expected result and how much that result means to the agent. All these factors are summarised in the expected payoff function described in Chapter 5. Therefore, the conditions of an agent's act selection, denoted by PfA, can be expressed as,

$$(7.33)\quad (\text{PfA}\ \alpha\ \varphi) \stackrel{def}{=} \left[\begin{array}{l} (\exists e((\text{Happens}\ e) \Rightarrow (\text{Intend}\ i\ \varphi\ \zeta)) \wedge (\text{Happened}\ e) \wedge \\ (\text{Bel}\ i\ ((\forall \alpha'(\text{Achieves}\ \alpha'\ \varphi)) \wedge (\exists \alpha\ (\alpha \subset \alpha')(\text{Prefer}\ i\ \alpha\ \varphi)))) \end{array}\right].$$

So, an agent's act selection can be formalised as,

(7.34) $\models (\text{Goal}\ i\ \varphi) \wedge \mathbb{A}(\text{PfA}\ \alpha\ \varphi) \Rightarrow (\text{Decide}\ i\ \alpha\ \varphi)$. That is,

> **Act selection:** An agent i has a goal φ. An expected event e, which enables the agent to hold an intention φ, has happened and the agent has a belief that among all the possible ways of bringing about φ, action α is the most preferable one. The agent then holds a mental state that decides to adopt action α as the way of bringing about φ.

It is worthwhile to notice the two possible cases in the goal definition (7.2). They are: 1) the agent can perform all the required actions α, ((Done α i); φ) to bring about φ. This means that the agent is capable of achieving φ by working in isolation. 2) The agent can only perform part of the required actions (α) but not all of them (α') ($\alpha \subset \alpha'$). It depends on other agents to perform certain actions (α'-α) to achieve the goal φ. However, the decision definition in this thesis (7.10) is based on the agent having an intention to bring about the goal φ. This means that when the agent makes a decision about which action to adopt, i.e. whether to work alone or in cooperation with other agents, the agent is aware of the possible cases and consequences. Another important point needs to pointed

out, is which the action denoted by α in the above act selection is different to the action, still denoted by α, in the goal definition. It could be the same action but it could be otherwise.

7.4.3 Team Formation

The team formation[8] in the SMM model is an action, as part of action α in the decision formula, that the agent has to perform after it decides to work in a cooperative manner. This action is to build up a common mental state among a group of potential cooperative agents towards a collective action. This mental state has been identified as *team.* It includes four distinctive features: 1) common act rules, 2) a common base of sharing resources, 3) a common measure of performance and 4) mutual mental state adjustment. Theoretically the above four features are summarised in the notion of *joint intention* (7.29) which is expressed by the notions of joint commitment and social convention. Joint intention as the goal of team formation cannot be guaranteed by the agent that performs team formation because this particular action depends on factors that are beyond its control such as the mental states of potential team members. The agent can only *attempt* it. The team formation stage is then characterised as an assumption made about rational agents. Namely, that an agent that decides to adopt cooperative action will solicit assistance.

> **Team formation:** An agent i, that decides upon a cooperative action with respect to its goal φ, will attempt to bring about in some group g that it believes can jointly achieve φ, a state wherein: (i) it is mutually believed in g that g can jointly bring about φ and g jointly intend to do so; or, failing

[8] Intuitively, team formation only happens when agents decide to work in collective actions. However, this thesis regards an agent working in isolation as a single member team. Obviously, if an agent decides to work in isolation, it may not need to consider the issues related with the team formation. Having this stage is serving the purpose of generalisation. Nevertheless, the theory is still valid for the agent work in isolation because when certain events happened, the agent may find that working in isolation is not favourable any more and turn to seeking assistance.

that, to at least cause in g (ii) the mutual belief that i has a goal φ and the mutual belief that i believes g can jointly achieve φ.

(7.35) $(\text{Team } g\ \varphi) \stackrel{def}{=} (\text{M-Bel } g\ (\text{J-Can } g\ \varphi)) \wedge (\text{J-Intend } g\ \varphi\ \zeta)$.

Therefore, their team formation can be stated as,

Assumption 1 (**Team Formation**):

(7.36) $\models \forall i\ (\text{Decide } i\ \alpha\ \varphi) \Rightarrow \mathbb{A}\exists g\ (\text{Happens } (\text{Attempt } i\ \alpha' \mathit{Team}\ q))$.

where, α' is team formation, $\alpha' \sqsubset \alpha$,

$\mathit{Team} \stackrel{def}{=} (\text{Team } g\ \varphi)$, and

$q \stackrel{def}{=} (\text{M-Bel } g\ (\text{Goal } i\ \varphi) \wedge (\text{Bel } i\ (\text{J- Can } g\ \varphi)))$.

7.4.4 Plan Formation

The fourth stage of the SMM model is plan formation. It is based on the belief that a collective action cannot begin until the group to some extend know about what they will actually do even when the team has been formed. Plan formation is a procedure that brings the team to a state, in which there is at least one action that an agent in the group can perform and it is known to the group that this will take them "closer" to the goal. However, it is possible that there are many agents that know of actions the team can perform in order to take the collective closer to, or even achieve the goal. It is therefore necessary for the collective to come to some agreement about exactly which course of action they will follow and which agent will play which part in this course of action. *Negotiation* is the mechanism via which such agreement is reached. The *negotiation* is formalised as a group of agents jointly attempting to bring about a state where they have agreed on a common plan, and decided to act on it. Failing that, they will bring about a

state where at least one of them has proposed a plan that is believed will achieve the jointly intended goal.

Negotiation: If group g are a team with respect to a goal φ, then g will jointly attempt to bring about a state (by performing α''), where it is mutually decided in g that adopting an action sequence α_p to achieve their jointly intended goal φ, or, failing that, to at least bring about a state where some agent $j \in g$ has made g mutually aware of its belief that some action α_p' can be performed by g in order to achieve φ.

That is,

Assumption 2 (Plan formation):

(7.37) $\models (\text{Team } g\ \varphi) \Rightarrow \mathbb{A} \exists \alpha'' (\text{Happens } (\text{J-Attempt } g\ \alpha'' (\text{Plan} \alpha_p\ \varphi)\ q))$

where, $(\text{Plan } \alpha_p\ \varphi) \stackrel{def}{=\!=} \exists \alpha_p\ (\text{J-Decide } g\ \alpha_p\ \varphi)$,

$q \stackrel{def}{=\!=} \exists j\ \exists \alpha_p' (j \in g) \wedge (\text{M-Bel } g\ (\text{Agts } \alpha_p'\ g) \wedge (\text{Achieves } \alpha_p'\ \varphi)))$.

7.4.5 Team Action

The stage of team action in the SMM model is an expectation that follows from the plan negotiation. In this stage, the team members will jointly attempt the actions in the fixed plan.

Team action: A group g as a team with joint intention to the goal φ, if there is a decided plan α_p, the team will jointly perform the actions in the plan under a "code of conduct"- joint commitment and social convention.

That is,

(7.38) $\models (\text{Plan } \alpha_p\ \varphi) \Rightarrow \mathbb{A}\ (\text{Happens } (\text{J-Attempt } g\ \alpha_p\ \varphi\ S\text{-}C_{JPG}))$.

7.4.6 Shifting

Shifting is the last stage in the SMM model. As described in the SMM framework, it indicates the termination of the cooperation in a team format. Shifting means the changing of agents' mental states since the attainment of the intended goal. The mental states include goals, intentions and beliefs. This change of mental states also causes the agents' behaviour to change, such as shifting their position in a matrix structure and renew their roles in the next goal attainment. Most importantly, the SMM framework defines the cooperation process as such that it is not a single-pass process, instead, it is a circular process. Rational agents will adjust and update their mental states in responding to their experience of success and failure in the interaction with other agents. Therefore, shifting is a process of learning.

> **Learning:** With respect to an unknown event e, an agent i has enough evidence or experience about it, the agent will perform an action to enable a belief about the event to be held.

That is,

$$(7.39)\quad (\text{Learn } i\ e) \stackrel{def}{=} \exists e((\text{Happened } e) \wedge ((\text{Agts } e\ g) \wedge (i \in g)) \wedge \neg(\text{Know } i\ e)) \Rightarrow \exists \alpha\, \mathbb{A}\, ((\text{Does } \alpha\ i)\ \mu\ (\text{Bel } i\ e)).$$

Therefore, the shifting can be expressed as,

$$(7.40)\quad \models \forall i\, ((i \in g)(\text{Happened } (\text{J-Attempt } g\ \alpha_p\ \varphi\ S\text{-}C_{JPG})) \Rightarrow (\text{Learn } i\ (\text{Done } \alpha\ i))).$$

Notice that the action α in the above (Done $\alpha\ i$) is the action that agent decided to adopt in act selection (7.34). All other actions in the other stages are part of it. The shifting formulation expresses the knowledge which is in a general sense that the agents in group g do not know, and should be learnt. The result of the agents' learning should have an impact on the agents' beliefs which further affects the agents' expected function $E\omega$ in

their future decision making.

7.5 Summary

In this chapter, an abstract formal model of SMM has been described against the three objectives stated in the beginning of the chapter. They were 1) to provide formulation of multi-agent cooperation in dynamic environment; 2) to produce a mathematically tractable theory; and 3) to offer a clear mapping from the theory to its applications. The formal model of SMM is described based on *possible worlds* semantics, *temporal logic* and *dynamic logic*. The language used in the description is a multi-model language with usual connectives of a first-order language. This language has been used in many articles in multi-agent systems theory, therefore only descriptive semantics were given.

Based on the previous experience of implementation, the basic attitudes and the necessary actions of an agent which are needed in a cooperation process have been specified in this chapter. The mental states, which govern the agent's individual and social behaviours, were defined as different events under *possible worlds* semantics. Doing so enables the relationship among the mental states of an agent and between the agent's mental states and actions to be studied and described.

Finally, following the SMM framework proposed in Chapter 5 a formal model of multi-agent cooperation was presented with six separate behaviour stages. This formal SMM model not only provides the agents' mental and physical behaviours in a cooperation process but also specifies the sequence and the relationship between the agents' behaviours. Therefore, it can be used for both theoretical and implementation purposes.

In summary, the following features of the SMM model are identified to distinguish it from other models against which its significance can be gauged.

1. The model carefully works out the mental states and actions of an agent that are needed for a cooperation process. The mental states of an agent include motive, capability, belief, goal, intention, preference, decision, mutual belief, mutual goal, joint decision and joint intention. The actions of an agent are done, doing, does, attempt, joint attempt and learn.

2. The model identifies the sequence and the relationship between an agent's mental states and actions not based on a randomly considered time but from the very beginning of the system where the agent can be viewed as existing. That is the motive and capability which are the identity of the agent. They are the initial states of the agent. All other mental states and actions of the agent are derived from these initial mental and physical states. It has a clear advantage for applications because the motive and capability can be designed and expressed with any computational language.

3. The model, for the first time, introduces notions to express an agent's mental states that can be quantitatively evaluated. That is the notions of deciding and its derived notions are based on a quantitative payoff function $E\omega$ which is further expressed by the quantitative functions ρ, υ and ϕ. Where ρ is the output function that states the agent's idea about outcomes of an action determined by the agent and the environment; υ is utility function that states how the agent ranks alternative outcomes of an action; and ϕ is the probability density function that states the probability of the event under which the expected outcome can be obtained.

4. The model resolves the argument against intention theory that claims it is invalid in agents' cooperative behaviour. This is done by explicitly specifying that the intention can only operate on agents' mental states, i.e. the goal. This means that the intention

only characterises an individual agent's focus on a specific goal; and the joint intention only characterises a group of agents' focus on a mutual goal. It is the decision that operates and characterises agents' act selection.

5. The SMM model provides a logic and implementation process that ensures cooperation can take place and can achieve success for multiple agents sitting in a dynamic environment.

It should be clear now that the formal SMM model, like many models in multi-agent systems is concerned with searching agents' basic attitudes and the relationships that govern agents' physical behaviours by using powerful logic tools and bearing rationality in mind. It also integrates many valuable research results from other fields such as organisation theory, economy, and cognitive science. Together these form the fundamental point of this thesis.

This is that agents, in any given time in a multi-agent system, inevitably find that they are located in a cross point which is jointly defined by their motive, capability, and the environment. Agents' action and the consequences of the action result in shifting their position from one point to another. This is why the model is called shifting matrix management.

Chapter 8

Experimental Evaluation of the SMM Model

Is it Gold or sand? It can be told by wild fire.

-- A Chinese proverb

This chapter reports on the empirical investigations of the SMM model. A number of controlled experiments were performed using the test-bed built based on ARBS, which was described in Chapter 2. Section 8.1, describes the implementation of the SMM model in the control of cooperation between multiple robots, which was carried out to study the effectiveness of the SMM model in the problem domain. In section 8.2, taking this application as an example, the systemic characteristics of the SMM model are analysed. Finally in section 8.3, a comparison is made between the SMM model and the two other models described in Chapters 3 and 4. A number of experiments with the

same system settings have been carried out. The main benefits of the SMM model are stated as testable hypotheses and experiments were undertaken to evaluate these claims. Although the scenario used to illustrate the concepts is that of multiple robots' cooperation, the results obtained are believed to be generalisable.

8.1 Implementation of the SMM Model

In this section, implementing the SMM model in the control of cooperation among multiple robots is reported. The implementation is parallel to the implementation of two other models, which were reported in Chapters 3 and 4. The logic developed in the previous chapter was realised by the rules and functions in different KSs in the ARBS. The KSs were organised according to the logic order defined by the SMM model. The implementation was carried out to investigate the applicability of the SMM model to the problem domain and to demonstrate how the logic provided by the SMM model can be converted into an application system.

8.1.1 Implementing the SMM Model

Implementing the SMM model in the control of cooperation among multiple robots is like implementing the Contract Net framework and the CPS framework in the same problem domain. It is based on the existing test-bed constructed in this research.

This time, the blackboard has been partitioned in the manner shown in Figure 8.1. The differences between the partitions shown in the Figure 8.1 and the partitions in Figures 3.2 and 4.1, which are implementations of the Contract Net framework and the CPS framework, are mainly revealed in the agents' private panels.

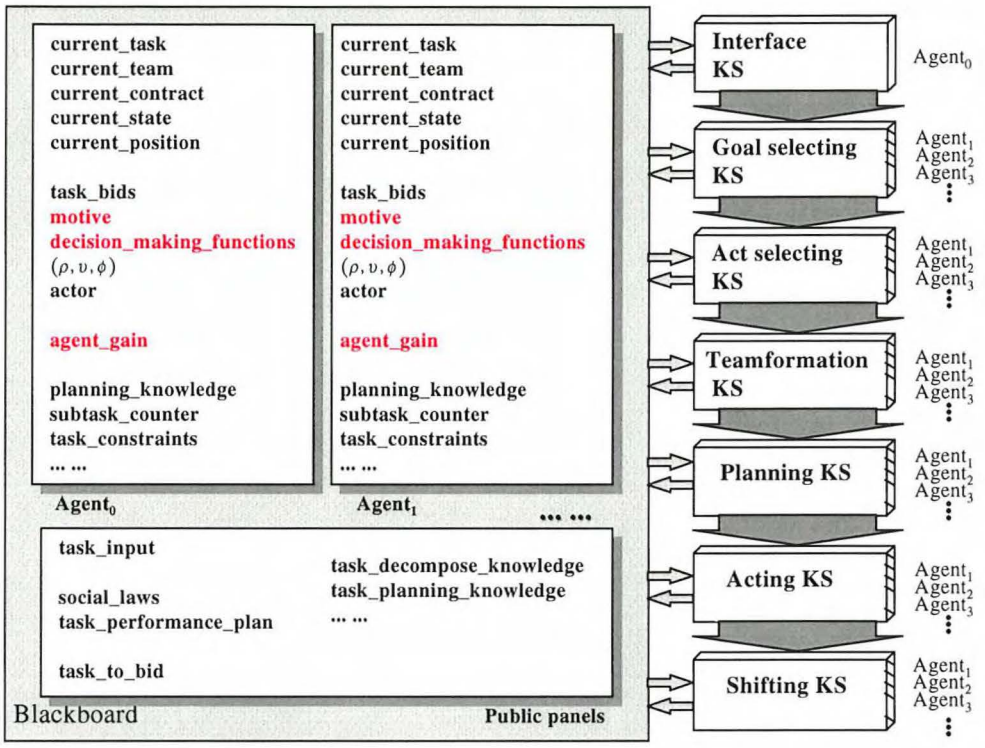


Figure 8.1 The blackboard partition for implementing the SMM framework

This partitioning establishes three new partitions in the private panel. The motive partition[1] is used to store the agent's initial motive. The motive is a proposition with explicit quantitative expression, such as to obtain 10,000 units of net benefit gain or to obtain as much benefit as possible. The decision_making_functions partition keeps the names of the three decision making functions ρ, υ, and ϕ. The actual body of the functions is written using ARBS' external function facility. The links between the name and the body of functions are realised by the ARBS *runalg* operator. The agent_gain partition records the benefits that the agent gained from the task's performance. It is used to check against the agent's motive to see if its motive has been achieved. These three private partitions were added because of the requirements of the SMM model.

The message format used in communication between the agents is similar to the format of the messages used in the CPS framework, which is illustrated in Figure 4.2. The only

[1] Partition is an element of a panel.

difference is the slot of team specification in the team proposal message. The SMM model does not only explicitly specify the "code of conduct", but also specifies the sharing of the resources, the same performance measurement, and the mutual adjustment of agents' mental states. Figure 8.2 shows the new format of the team proposal message.

Type: team proposal

From: $agent_0$

To: task_to_bid (a broadcast message)

Team: 01

Task Abstraction:

task type: performing

task: move [block1 block3 block4] from position2 to position7

task weight: 40, 10, 120, 20.

Team Specification:

1. Act under joint commitments and social conventions.
2. Sharing resources.
3. Performance is measured by the same standard.
4. Mental states need to be mutually adjusted.

Figure 8.2 The structure of the team proposal message

The KSs and their order are similar to those used in the implementation of the CPS framework. Because of the differences between the SMM model and the CPS framework, two new KSs were added. They are act selecting KS and shifting KS. Rules in the act selecting KS realise the decision theory on alternative actions. The quantitative functions were realised by ARBS' external functions, written in C. The rules in shifting KS realise the learning process and the mutual adjustment of agents' mental states. Other behaviours governed by different mental states of agents in the SMM model require changes to the rules in the KSs used in the CPS framework. The new rules were designed to follow the logic defined in the SMM model. The KSs and the order of the KSs used in implementing the SMM model are shown in Figure 8.3.

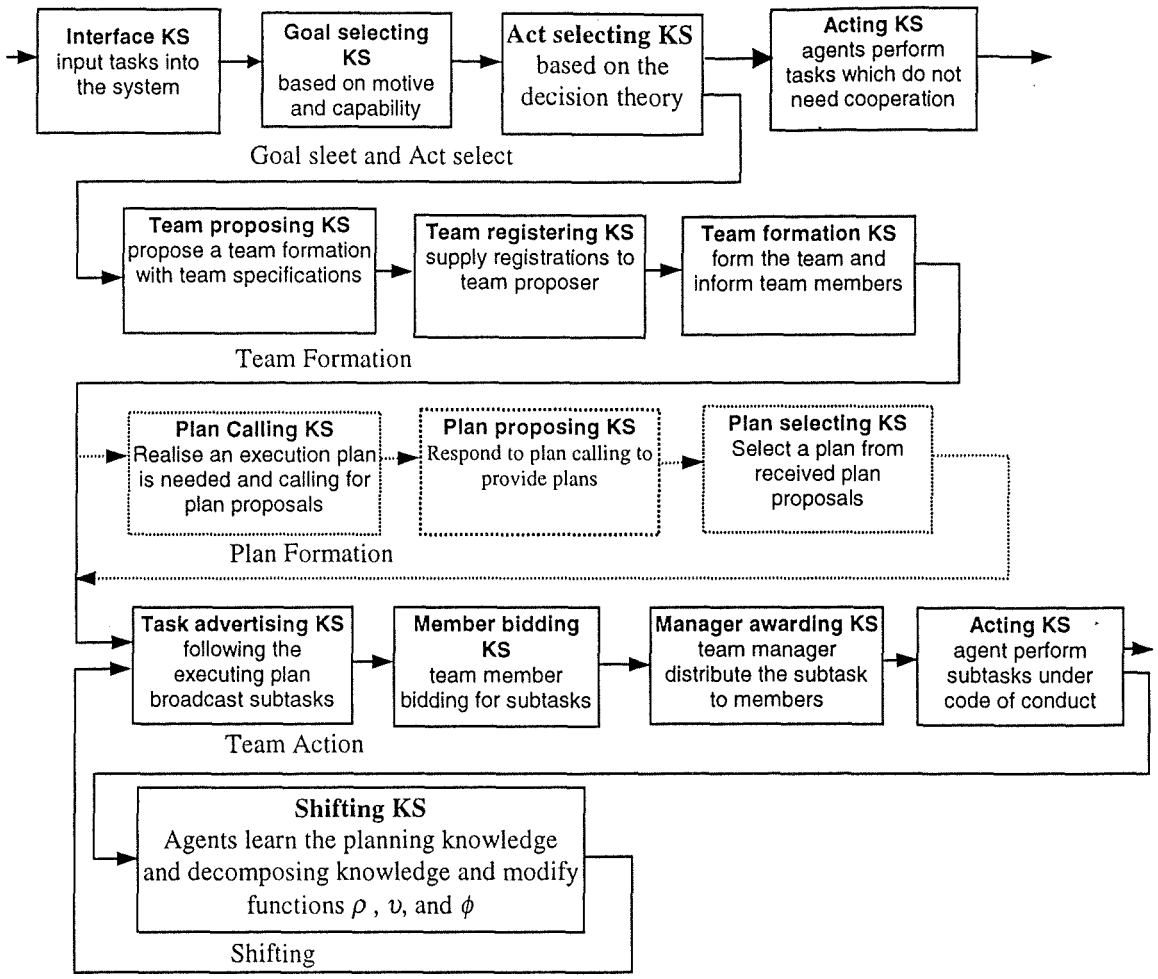


Figure 8.3 The KSs and their order in implementing the SMM model

8.1.2 Tests, Results and Remarks

To test the applicability of the SMM model, the tasks, which were generated from the application domain and used to test the applicability of the Contract Net framework and the CPS framework, have been modified. Each task has a weight associated with it. The weight of a task is four quantitative parameters in the following order: *social benefit*, *social cost*, *self benefit* and *self cost*. These parameters are assumed to be defined by the users to address the nature of the importance and the difficulty of a task. They are not the value that an agent associates with the task. However, these values are the basic factors that an agent uses to evaluate the tasks. For the testing purpose, the tasks' weights were assigned arbitrarily with integers between 0 and 200, which is illustrated in Table 8.1.

Table 8.1 The tasks and their weights

Atomic Tasks	Weight	Combined Tasks	Weight	Complex Tasks	Weight
Park1	10, 0, 50, 20	Hide	90, 30, 70, 40	Move(front, back)	140, 60, 80, 40
Bend_over	10, 0, 60, 20	Lay_on _left	100, 20, 50, 10	Move(left, right)	140, 60, 80, 40
Lay_down	20, 10, 60, 20	Lay_on_right	100, 20, 50, 10	Move(left, back)	140, 60, 80, 40
Sit_down	20, 10, 60, 20	Lay_on_front	100, 20, 50, 10	Move(front, right)	140, 60, 80, 40
Left	20, 15, 50, 10	Lay_on_back	100, 20, 50, 10	Move(position1, position4)	160, 60, 100, 20
Right	20, 15, 50, 10	Sit_left	100, 30, 60, 10	Transport(inter_front,	160, 70, 100, 40
Front	20, 15, 50, 10	Sit_right	100, 30, 60, 10	inter_back)	
Back	20, 15, 50, 10	Sit_front	100, 30, 60, 10	Transport(inter_front,	160, 70, 100, 40
Grip	10, 0, 60, 10	Sit_back	100, 30, 60, 10	inter-right)	
Release	10, 0, 60, 10	Left_grip	90, 20, 70, 25	Transport(inter_left,	160, 70, 100, 40
Gripper_up_down	10, 0, 60, 10	Left_release	90, 20, 70, 25	inter_back)	
Gripper_sideway	10, 0, 60, 10	Right_grip	90, 20, 70, 30	Transport(inter_left,	160, 70, 100, 40
Inter_left	50, 20, 50, 10	Right_release	90, 20, 70, 30	inter_right)	
Inter_right	50, 20, 50, 10	Front_grip	90, 20, 70, 30	Hanoi_tower(3, 5)	200, 40, 200, 60
Inter_front	50, 20, 50, 10	Front_release	90, 20, 70, 30		
Inter_back	50, 20, 50, 10	Back_grip	90, 20, 70, 30		
Position1	80, 30, 50, 30	Back_release	90, 20, 70, 30		
Position2	80, 30, 50, 30	Hand_over	120, 40, 60, 20		
Position3	80, 30, 50, 30	Hand_out	60, 20, 30, 10		
Position4	80, 30, 50, 30	Hand_in	60, 20, 30, 10		

The agent's value assessment of a task depends on the motive of the agent, the output function ρ and the utility function υ that the agent adopted. In this implementation the motive of agents was assigned as "to gain as much benefit as possible." The three quantitative functions were defined in a simple form[2]: the output function ρ and the utility function υ together, which has been defined as a payoff function ω (see Chapter 6), were defined as the sum of a task's net benefit[3]. For example, the task *move (front, back)* has been weighted (140, 60, 80, 40), the net benefit of performing the task is (social benefit - social cost) + (self benefit - self cost), which is (140 - 60) + (80 - 40) = 120. The probability density function ϕ, was defined as a value associated with each individual agent. If an agent works alone, it is regarded as a team only consisting of the agent, the probability of teamwork in this case is 1 because the agent is sure that teamwork can take place. A mean arbitrary value of 0.5 was taken as the initial value for an agent to work with other agents since it is assumed that the agent has no experience of working with the other agents. This is because the agent regards the possibility of teamwork as having an even chance of success. This value was altered by an arbitrary value 0.01 each time, responding to the agent's own experience of success or failure. With each success the agent experiences with another agent, the probability value of teamwork with that agent will be increased by 0.01. Similarly, if the agent experiences a failure from teamwork with that particular agent, the agent will decrease the value of probability of the teamwork with that agent by 0.01. The final value of a probability, however, cannot be less than 0 or greater than 1. The probability of teamwork, where more than one agent is engaged in a team, was calculated by taking the mean value of the probability from every agent in the team. A non-linear function could be used as an

[2] Different functions and their impact on the system performance are studied in the next section.

[3] Here an assuption was taken that an agent values the importance of performing a task to itself is in the same order as one defined by the task's net benefit.

agent's probability function, however, for the sake of simplicity this implementation adopted the above linear probability function. For example, when an agent *i* evaluates the chance of teamwork with agent *j*, if the agent *i* has had one success from teamwork with agent *j*, the value of the probability of working with agent *j* will be increased by 0.01 from the initial value 0.5, it becomes 0.51. If agent *i* had two successful experiences with agent *j* and one failure with agent *k*, agent *i* will using the value of $(0.5 + 2 * 0.01 + 0.5 - 0.01) / 2 = 0.5033$ as the probability of teamwork which the team has three members: agent *i, j, and k*. It is clear now that the above simple definitions about ω and ϕ match up with the "bench marking" scheme in the quantitative decision theory. After all an agent makes a decision between two alternative actions α and α', it will simply calculate the expected payoff of the two actions by using the expected payoff function, $E\omega(\alpha) = \omega(\alpha) * \phi(\alpha)$, and compare the results, and adopt the action with the highest value.

In the tests, six agents were used to represent two robots with different position coverage and physical functionality, which remain the same as for the tests of the Contract Net framework and the CPS framework (see Table 3.4). Each agent has an identical structure, which was described in Chapter 2 and implemented in ARBS. The agents had sufficient computational power, but vary in their inbuilt knowledge, which is illustrated in Table 3.4. The 50 generated tasks, which were used to test the Contract Net framework and the CPS framework, were tested. The results are shown in Table 8.2, which demonstrates the accomplishment of all the tasks. The applicability of the SMM model in the control of cooperation among multiple robots has been illustrated.

Table 8.2 The results of tests

Atomic Tasks	Status and performer	Combined Tasks	Status and performer	Complex Tasks	Status and performer
Park1	Done. Robot1.	Hide	Done. Robot1.	Move(front, back)	Done. Robot1, Robot2.
Bend_over	Done. Robot1.	Lay_on _left	Done. Robot1.	Move(left, right)	Done. Robot1, Robot2.
Lay_down	Done. Robot1.	Lay_on_right	Done. Robot1, Robot2.	Move(left, back)	Done. Robot1, Robot2.
Sit_down	Done. Robot1.	Lay_on_front	Done. Robot1.	Move(front, right)	Done. Robot1, Robot2.
Left	Done. Robot1.	Lay_on_back	Done. Robot1, Robot2.	Move(position1, position4)	Done. Robot1, Robot2.
Right	Done. Robot2.	Sit_left	Done. Robot1.	Transport(inter_front, inter_back)	Done. Robot1, Robot2.
Front	Done. Robot1.	Sit_right	Done. Robot1, Robot2.		
Back	Done. Robot2.	Sit_front	Done. Robot1.	Transport(inter_front, inter-right)	Done. Robot1, Robot2.
Grip	Done. Robot1.	Sit_back	Done. Robot1, Robot2.		
Release	Done. Robot1.	Left_grip	Done. Robot1.	Transport(inter_left, inter_back)	Done. Robot1, Robot2.
Gripper_up_down	Done. Robot1.	Left_release	Done. Robot1.		
Gripper_sideway	Done. Robot1.	Right_grip	Done. Robot2, robot1.	Transport(inter_left, inter_right)	Done. Robot1, Robot2.
Inter_left	Done. Robot1.	Right_release	Done. Robot2, robot1.		
Inter_right	Done. Robot2.	Front_grip	Done. Robot1.	Hanoi_tower(3, 5)	Done. Robot1, Robot2.
Inter_front	Done. Robot1.	Front_release	Done. Robot1.		
Inter_back	Done. Robot2.	Back_grip	Done. Robot1, Robot2.		
Position1	Done. Robot1.	Back_release	Done. Robot1, Robot2.		
Position2	Done. Robot1.	Hand_over	Done. Robot1, Robot2.		
Position3	Done. Robot2.	Hand_out	Done. Robot1, Robot2.		
Position4	Done. Robot2.	Hand_in	Done. Robot1, Robot2.		

Perhaps what is important and needed to draw attention to is that the system has an incremental performance. This was obtained by inputting the testing samples repeatedly in the system in different rounds. The four rounds of the test were recorded and their statistic data are presented in Figure 8.4. Where, the y-axis represents the tasks, which have been completed by the system, and the x-axis represents the time unit (in seconds) taken in accomplishing the tasks.

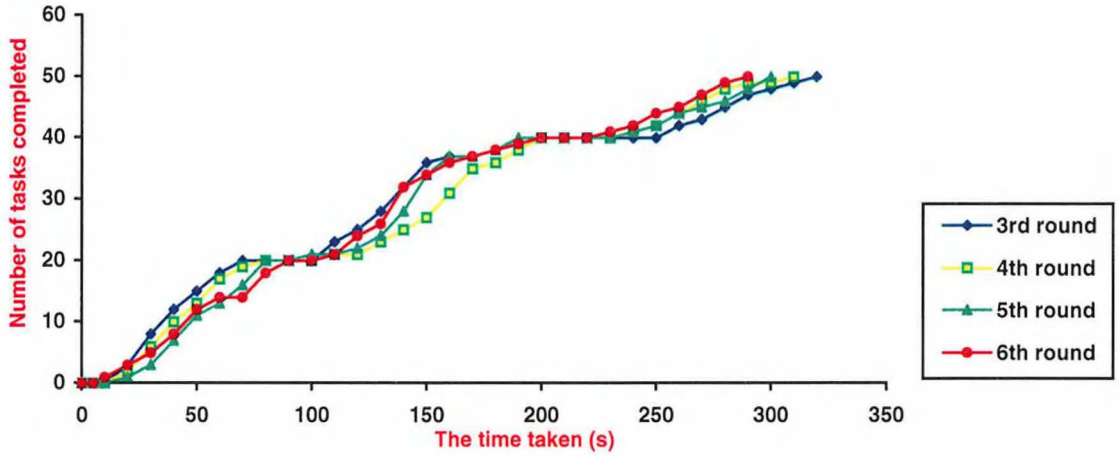


Figure 8.4 Tasks accomplishment in different round of the tests

Notice that the figure shows rounds three to six. This is to avoid statistical anomalies associated with start-up. After six rounds the system appeared to show no significant improvement and so was regarded as being stable. The figure shows that, in the sixth round the time to accomplish tasks has been reduced by almost 30 seconds compared with the third round of the test, which is about 10 percent in total. That means the system performs more efficiently in round six than it did in round three. The time saving mainly arises in the agents' decision making and the team formation attempt. This is because of the agents' mutual adjustment and learning from their experience of success and failure of teamwork in the past.

Notice the assumptions made in the implementation of the SMM model in the control of cooperation between multiple robots. Firstly, the motive of the agents has been defined

as "to gain as much benefit as possible." This means that the agent will select to perform the tasks that it is able to perform. It is no different to the benevolent assumption used in the CPS framework, which was stated in Chapter 4. Secondly, the definition of the agents' expected payoff function mean the agents will never attempt to work cooperatively, if it is possible to work in isolation. This is because the payoff of working in isolation ($\phi = 1$) is always higher than the payoff of any teamwork ($\phi < 1$) if both working in isolation and teamwork can achieve the selected task's performance. This was done because the author believes that in multi-robot's domain certainty (or risk) should have higher priority than efficiency. However, in a problem domain where the priority between certainty and efficiency may be altered, or under same certainty, efficiency may become the main focus, this can be done by re-defining the payoff function ω in $E\omega$ and making the efficiency of teamwork greater than that for working in isolation. Thirdly, there is no difference in the individual agents' motive, and expected payoff functions. A lack of variety in the agents' initial setting results in similar agent's behaviour in the system.

The above assumptions restrict the full potential of the SMM model. In practice the agents in a system may be different in capability, such as in a multiple robots' domain, where we can expect different functionality, since the robots are made by different companies at different times, and the control mechanisms, and the technology involved are also different. To represent this variety, the agents should possess different motives and decision functions. To fully explore and analyse the potential of the SMM model, a detailed study of the performance of the SMM model is presented in the next section.

8.2 Varying the Performance of the SMM Model

In this section, the performance the SMM model is analysed by means of implementing

the SMM model in the control of cooperation among multiple robots. The initial settings of the system, which are defined by the SMM model, such as the motives of the agents, the decision functions that agents' adopt and the weight of the tasks, are treated as variable parameters. The systemic performance is then investigated with different settings.

8.2.1 The Settings of the Experiments

The SMM model presented in this thesis identifies two distinctive mental states that affect an agent's behaviour and the behaviour of the overall system. One is an agent's motive, which directs the agent's task selection. The other is the expected payoff function that agents adopt, which affects the agent's act selection. These two mental states are variables, which need to be set up in the initial configuration of the system. The motive and the decision functions actually reflect a notion called *personality* in the multi-agent systems research [Sloman 97]. An agent's personality refers to the difference between agents in both mental and physical behaviours that can be identified to distinguish each individual agent. The four types of commonly occurring agent in multi-agent systems are:

- Benevolent agent: Agents that perform all the tasks that they are capable of. This agent type is dominant in the reductionist view of multi-agent systems and has been widely applied in industrial robot systems.

- Selfish agent: agents that only choose and perform tasks that bring them a positive net self benefit. This agent is dominant in the constructionist view and widely used in software agent systems.

- Selfless agent: agents that only choose and perform tasks that will bring a net benefit to the society they belong to. These agents are not concerned with their

individual benefits and losses.

- Socially responsible agents: agents that only choose and perform tasks if their benefits for society and for agents together is greater than the total cost to society and to agents.

The corresponding motive and decision functions to these four types of agents are defined as follows:

1. Motive definitions. The definition of the motive corresponding to different types of agents is listed in table 8.3. The definitions are coherent with the agents' personality definitions. This is because in the SMM model motive is used to select tasks. It is the same as the definition of the agents' personality.

Table 8.3 The definitions of the motive corresponding to agent's type

Type of agent	Motive
Benevolent	All the tasks that it is capable of
Selfish	Self benefit > Self cost
Selfless	Social benefit > Social cost
Socially responsible	Social benefit + Self benefit > Social cost + Self cost

2. Definitions of the payoff functions. The definition of the payoff function reflects the personality of an agent applied to different actions in order to achieve the agent's intended goal. The problem, which arises on this occasion, is how to define the action's weight. In this set of experiments the weight of an action is defined by the task and the number of actions to realise the task. Precisely, the weight of a task is evenly distributed into actions that realise the task. For example, the task *moving (front, back)* is weighted by a list (140, 60, 80, 40) and suppose it is decomposed into two actions: action1 *move from front to position2* and action2 *moving from position3 to back*, where position2 and

position3 are the same position in different robots' coordinate system. Action1 and action2 are then weighted (70, 30, 40, 20) respectively. The agent will select its next act always with what it considers, according to its personality, to be the highest value. This is because the agent regards the action with the higher value to mean more to it than the action with the lower value.

3. Probability function definition. The probability function in the SMM model expresses an agent's own opinion about the possibility of an event that can take place and can achieve an expected result. Therefore it is related to the other agents that are involved in the same event. The definition of the probability is defined in three dimensions: the initial value of the probability that the agent has given to other agents, the number of agents involved in the event, and past experience. The initial value of the probability reflects the agent's initial opinion on other agents. It has been defined in Table 8.4.

Table 8.4 The initial value of the probability definition

Assignor / Receiver	Benevolent agent	Selfish agent	Selfless agent	Socially responsible agent
Benevolent agent	0.5	0.5	0.5	0.5
Selfish agent	0.4	0.5	0.4	0.4
Selfless agent	0.5	0.5	0.5	0.5
Socially responsible agent	0.5	0.5	0.5	0.5

Notice that the initial values of the probability other agents give to selfish agents is less than to other agents. This is because the selfish agent has more chance not to join teamwork since teamwork requires social behaviour such as the behaviour defined by *joint commitment* and *social conventions* outlined in Chapter 7. This value can be changed in other implementations. This definition is only for the purpose of these experiments. The experience represents the agent's adjustment on its initial probability definition. As explained, each success or failure will increase or decrease the value of

the initial probability by 0.01. The final value of the probability is the average of the agents' probability and past experience. For example, a benevolent agent *i* evaluates the probability of an event that consists of four actions. These four actions are assumed to be performed by three agents *j, k,* and *l*. Agent *j* is a selfless agent and agent *i* has two success experiences with it; agent *k* is a selfish agent and agent *i* has no experience of working with it; agent *l* is another benevolent agent and agent *i* has one failure experience with it. Therefore the value of the probability that agent *i* holds for the event is: $[(0.5 + 2 * 0.01) + 0.4 + (0.5 - 0.01)] = 0.467$. Again this definition is simple but it follows conventional probability rules.

With the above motive and expected payoff function definitions, a number of controlled experiments were performed. In each experiment, four agents were used. They have the same basic architecture as shown in Chapter 2 and were implemented in rules and functions in the ARBS. In these experiments, two main dimensions were varied: the load on the system and the task's weight. This led to four types of experiment:

- The system has a heavy task load. New tasks are constantly inputted into the system. The agents always have tasks to perform.
- The system has a light task load. Significant gaps exist between the input of new tasks. The agents are often idle, as they have no tasks to perform.
- The social benefit is dominant. In the first two experiments, approximately equal weighting is assigned to the tasks for society's benefit and the individual's benefit. These experiments examine what happens when the social benefits significantly outweigh the individual benefits.
- The individual benefit is dominant. In these experiments the individual benefit

assigned to the tasks significantly outweighs the social benefit.

In the graphs in the following sections, the unit of time is seconds. To avoid statistical anomalies associated with start-up and termination conditions, the results are taken from the middle period (between 20s and 600s). In all cases, the time taken to complete a task varies between 1 second to 5 seconds. The y-axis shows the tasks that were completed by the agents.

8.2.2 System with Heavy Loading

In this set of experiments, the systems are heavily loaded so that the agents in the system always had tasks to perform. The weight assigned to each task is in a comparable scale. That means the net social benefit of a task is approximately the same as the net self benefit of the task. The value of the net benefits, both social and individual, can be positive and negative. Figure 8.5 shows the tasks completed by the different settings.

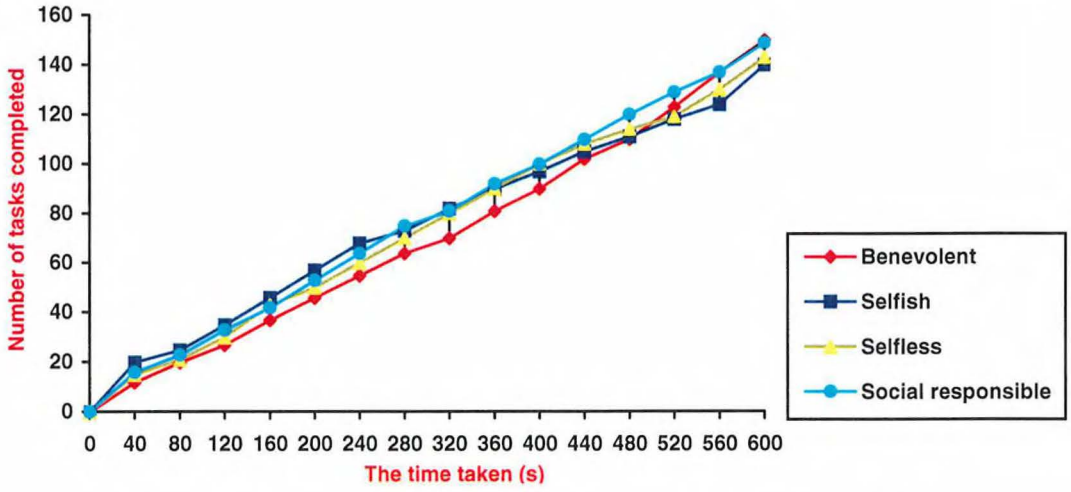


Figure 8.5 The tasks completed by the systems with heavy load

The figure indicates that the tasks, completed among the four settings in the system, are statistically almost the same. The maximum difference, at time 320 and 560, is less than 10 tasks. It demonstrates the coherent behaviour of the agents with different personality settings has been maintained by the SMM model. In other words, it indicates that the

SMM model has a great tolerance towards agents' different personality settings in systems with heavy loading.

However, this graph also shows that personality of agents indeed has an impact in these circumstances with regards to their different behaviour. The benevolent agents setting had a poor performance at the beginning of the experiments and the performance is gradually improved. It is revealed that in the beginning the line (red), which represents the benevolent agent setting, is below the lines representing other agents settings and in the end it rises above the other lines. In contrast, with the selfish agents setting, the system performs comparatively well at the beginning and the improvement tends to be small (blue line). The selfless and the social responsible settings are more stable than benevolent and selfish settings. This is shown by the smoother plots than that of the other two settings. It may indicate that systems with selfless and social responsible agents settings have less temporary difficulty than the other two settings. The difficulty, which the systems with selfish and benevolent settings have, is more failure attempts in the teamwork.

Nevertheless, the graph suggests that there is little difference between the personality types under these conditions. It is possible that the system is so highly loaded that any settings the agents in the system always have tasks to perform and are never idle.

8.2.3 System with Light Loading

In this set of experiments, the systems are lightly loaded. The agents in the system sometimes have no tasks to perform.

Figure 8.6 indicates that the system with the benevolent agents setting performs more tasks than the systems with other settings. But the actual tasks completed are

substantially less in magnitude, typically less than half of its counterpart shown in Figure 8.5.

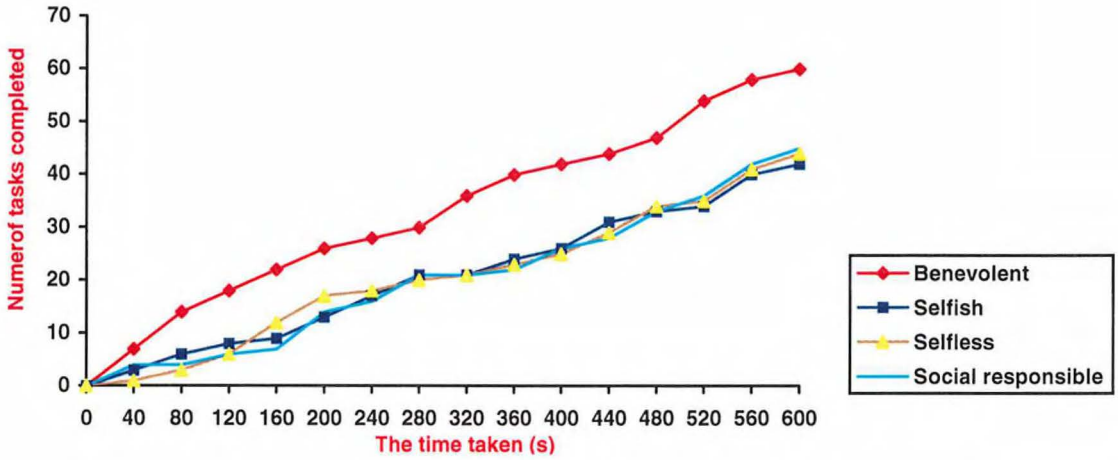


Figure 8.6 The tasks completed by the systems with light load

Figure 8.6 also indicates that all four plots are less smooth than the plots in the systems with heavy load. This is because at times the agents have no tasks to perform. In these circumstances, in the SMM model, the personality types of agents tend to have a greater impact than the system with heavy task loading. This set of experiments suggests that when implementing the SMM model in an application, if the tasks to the system are not heavily loaded then the benevolent setting may be a better choice.

8.2.4 Tasks with High Social Benefit

Having seen the effect of different tasks loading, the final set of experiments explores the system properties when the balance between the benefits to the individual and to the system is varied. In this case, the systems have a medium loading. In this section, the effect of making the social benefit two times greater than that of the individual benefit is examined. The net social benefit of tasks was given a range between ±1000, whereas the net individual benefits of tasks range between ±10. With this tasks' weight assignment, systems with different agent settings will behave significantly differently. These experiments aim to examine how the SMM model maintains the system's overall

performance. In these experiments the joint benefit, which is the net social benefit plus net individual benefit, is also monitored. This is because the joint benefit represents a balance between the individual agents and the society of the agents.

In Figure 8.7, the y-axis represents the joint benefits obtained by the system at the time it is measured. The figure shows that the systems with socially responsible and selfless agents' settings obtained much more joint benefit than the systems with benevolent and selfish agents' setting. The system with the selfless agents' setting almost obtains the same joint benefit as the system with the socially responsible agents' setting. This is because the joint benefit is virtually equivalent to social benefit since the social benefit overwhelms the self benefit by a factor of 100. The systems with benevolent and selfish agents' settings appear unstable and obtain less joint benefit. This is because the agents in these two systems sometimes perform tasks that have a high negative social benefit since the agents are not concerned with the social benefit. It suggests that, in a system in which the tasks' weight setting is dominated by the social benefit, the socially responsible or selfless agents setting will have less difficulty in performing social actions.

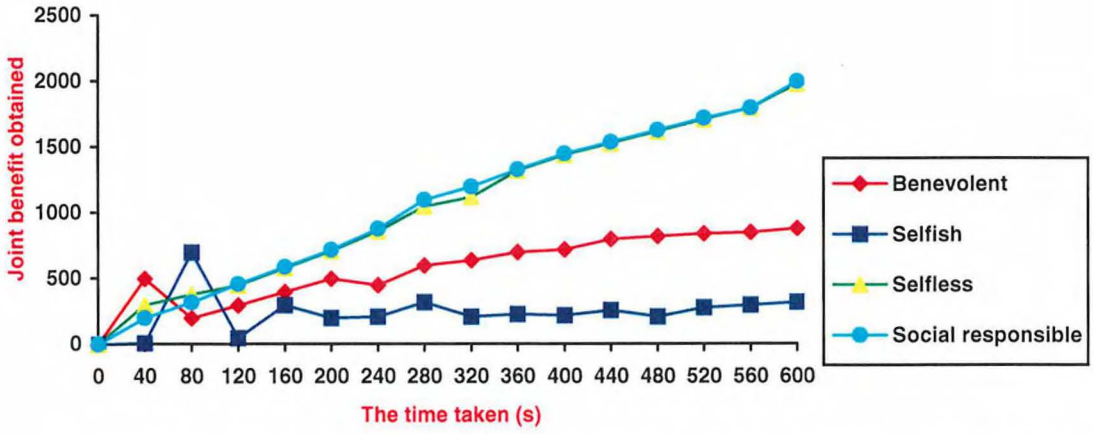


Figure 8.7 The joint benefit obtained by system with high social benefit

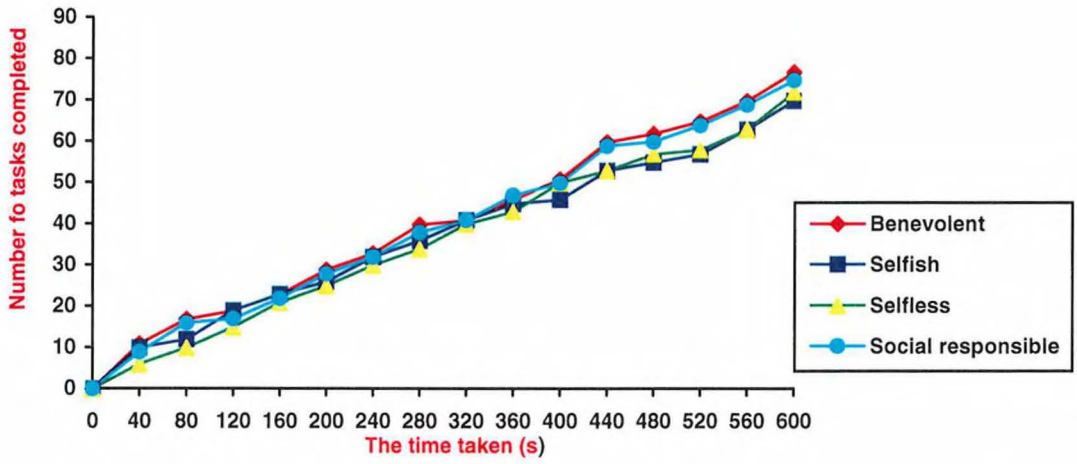


Figure 8.8 The tasks completed by the systems with high social benefit

What is surprising is that the tasks completed by the systems with different settings in this tasks' high social benefit setting are not affected very much by personality, see Figure 8.8. This is because the SMM model allows agents in the systems only to compare the differences between the available tasks according to their own concerns. While there are tasks to be selected the agents will always find a task to perform. They may find difficulty in the social actions such as team formation (the plots in Figure 8.8 are less smooth than in Figure 8.5), but the load of tasks is heavy enough to keep agents busy. The tasks completed by the systems can maintain a certain level[4].

8.2.5 Tasks with High Individual Benefit

In this set of experiments, the individual benefit of the tasks' weight is the dominant factor. It ranges between ±1000 whereas the social benefit ranges between ±10. Figure 8.9 shows the joint benefit obtained by the systems with different agents' settings. As might be expected the systems with the selfish and the socially responsible agents' settings obtain more joint benefit than the systems with the benevolent and the selfless agents' settings.

[4] Notice that we are not concerned with the task completion rate here. This is because the experiments set tasks to be continually inputted to the system. The tasks completed by the system are the factor that is of concern.

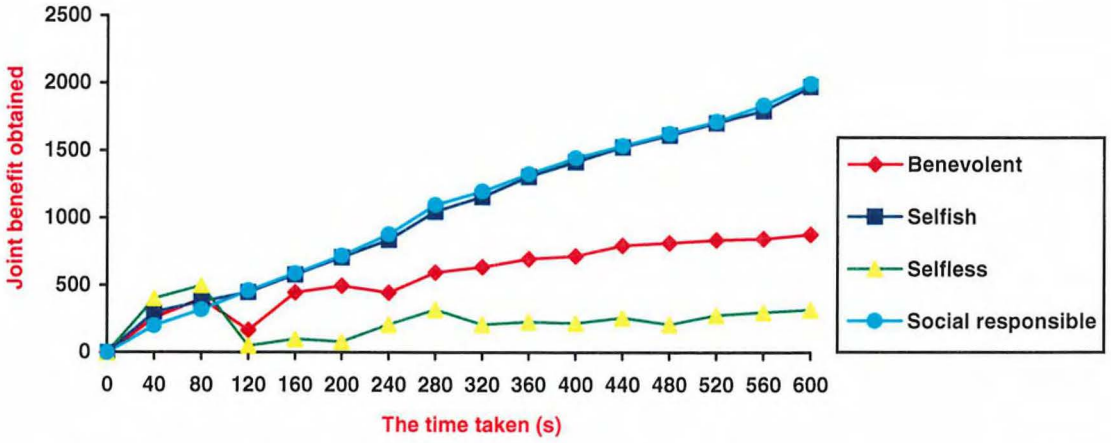


Figure 8.9 The joint benefit obtained by system with high self benefit

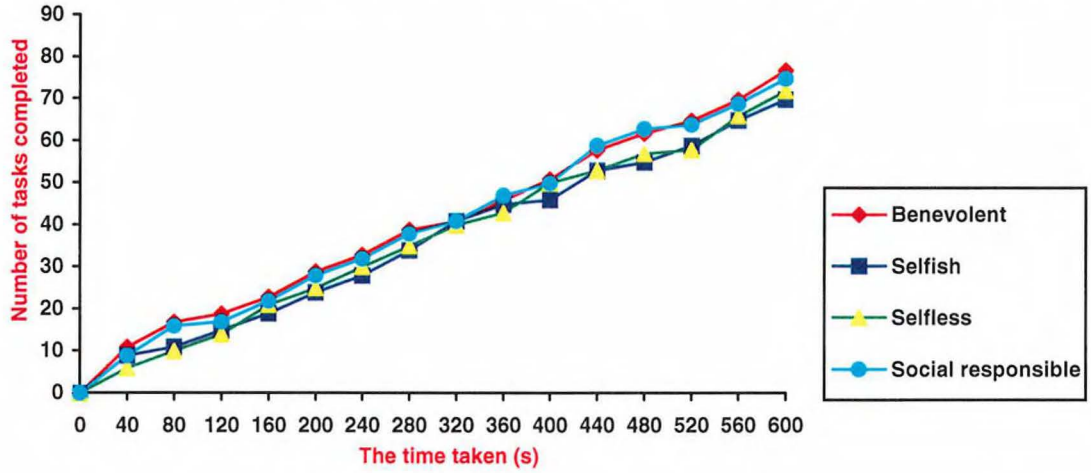


Figure 8.10 The tasks completed by the systems with high self benefit

This further demonstrates the result obtained from the previous experiments. That is, the tasks completed by the systems are not influenced by the tasks' weight setting if the tasks' load to a system is heavy enough to keep agents busy in the system. Also, the tasks completed by the systems with different agents' setting tend to be not significantly different either. Only the system with benevolent agents' setting tends to complete more tasks than the systems with other agents' settings, but not dramatically so.

8.2.6 Summary

In this section, to test the characteristics of the SMM model, four commonly occurring types of agents in multi-agent system were introduced. The motive and the decision functions were defined corresponding to these four types of agents. The different

systems were then set up in four sets of experiments. These varied in two dimensions: the task load and the tasks' weight setting. The results indicate that the SMM model performs well in maintaining coherent system behaviour with different settings. This is shown by the result that the tasks completed by the system are mainly dependent on the task load to the system rather than the agents' setting and tasks' weight setting. Nevertheless, the tasks' weight setting and the agents' type setting also have impact on the system behaviour where the SMM model is implemented as the systems implementing other models. The tests' results also suggest that if its agent's type settings match with its tasks' weight setting, the SMM system will have less difficulty in cooperative actions. In any case, a system with the benevolent agents' setting performs comparatively well in the sense of keeping agents busy. This may explain why the CPS framework assumes that agents should be benevolent. But it may not be a good choice in domains with scarce resources.

8.3 Comparison of the Cooperation Models

In this section, a comparison between the SMM model and two other cooperative models that have been implemented during the development of the SMM model is reported. In Chapter 3, this thesis claimed that the SMM model could be regarded as a refinement of the other models. This section investigates this claim.

Comparing one cooperative model with other cooperation models is a difficult task since each model has its own innovatory motivations, its particular domain of application, and particular goals to achieve. Ignoring these specifics in each model may invalidate the comparison. The comparison reported in this section is focused on a particular application domain, that is the control of cooperation between multiple robots. A series of controlled experiments were taken to evaluate the main benefits of the SMM

model that were taken as testable hypotheses. These experiments were not intended to provide a conclusive general judgement on each model since the experiments were not designed to fully explore each model's performance.

8.3.1 Experimental Hypotheses and Conjunctures

As stated in the beginning of Chapter 3, the SMM model can be regarded as a refinement of the existing cooperative models. To evaluate the result of the refinement, the following testing hypotheses were generated and as the objectives of the tests, and they are re-stated here:

1. The SMM model increases the task completion rate.
2. The SMM model increases the scope of achievable tasks.
3. The SMM model decreases the interference of task performance by avoiding harmful interaction.

To test these hypotheses, the task set that was used to test the three models has been doubled into 100 tasks. The time needed to complete an atomic task ranges between 1 second and 4 seconds. The combined tasks consist of between two and six atomic tasks. The complex tasks can be completed by different plans, which range from two to sixteen atomic tasks. Six agents were involved in each system. Three systems were constructed based on the Contract Net framework, the CPS framework, and the SMM model.

The system implements the Contract Net framework with one task manager agent[5] and six other agents. The system was organised as outlined in Chapter 3. The agents purely rely on the task's specification and their ability to select tasks to perform.

[5] The system implementing the Contract Net actually has one more agent that is the general task manager. But the general manager is not involved in the actual performance of any task. The actors that perform the tasks is still six.

The system implements the CPS framework with six benevolent agents. Two of them prefer teamwork (as part of cooperative conditions specified in the *Recognition* stage of the CPS framework). This means that whenever there is a goal to achieve the agents will attempt to achieve the goal by teamwork. In contrast, the other four agents only attempt teamwork when they are unable to achieve the goal. The system was organised as outlined in Chapter 4, in which no compromised methods were taken to overcome the problems perceived during the implementation. It can be regarded as truly reflecting the CPS framework in the problem domain.

The system implements the SMM model with six agents. Among them, two are benevolent agents, two socially responsible agents, one selfish agent and one selfless agent. The system adopts the implementation described in the first section of this chapter.

To make the results comparative, all the agents had the same structure as outlined in Chapter 2. Their physical function and the coverage of the working space were identical. Each agent was given an equal time slice and could carry out identical amounts of reasoning about the process of cooperation. The tasks' weight was kept same as stated in Table 8.2. Only the agents in the system that implements the SMM model made use of the tasks' weight, the agents in other systems ignored this information.

8.3.2 Task Completion Rate and Scope

To test the task completion rate, 100 generated tasks of the three types have been randomly mixed and continuously inputted to the systems. Figure 8.11 shows the task completion rate and the scope of the completed tasks. The data shown in the figures are the average of four rounds of separate tests. In the figure the y-axis represents the tasks completed by the systems, and the x-axis is the time in seconds taken to complete the tasks by the different systems.

Figure 8.11 shows that the system implementing the SMM model increases the task completion rate by 17 percent over the system implementing the CPS framework and 29 percent over the system implementing the Contract Net framework. It reaches an average of 5 seconds per task, compared with the average of 6.03 seconds per task for the CPS framework and 7.04 seconds per task for the Contract Net framework. It is hardly surprising, since this is what the SMM model is designed for. What is surprising is that both the systems with the Contract Net and the CPS framework have not completed the tasks as expected. The Contract Net system failed a few times when robots blocked each other's paths while performing tasks. Because of a lack of a mechanism for a social "code of conduct" in the Contract Net framework, the agents do not report the failure to perform the bidded task to the task manager, so the incomplete task is unknown to the task manager. It therefore does not re-advertise for agents to perform the task. In the CPS system, the tasks' performance failure occurred when one agent who prefers cooperative work finds another agent that also prefers cooperative work. Instead of working as a team, the second agent also proposes a team formation and the two agents become engaged in a loop, and the task is not performed[6].

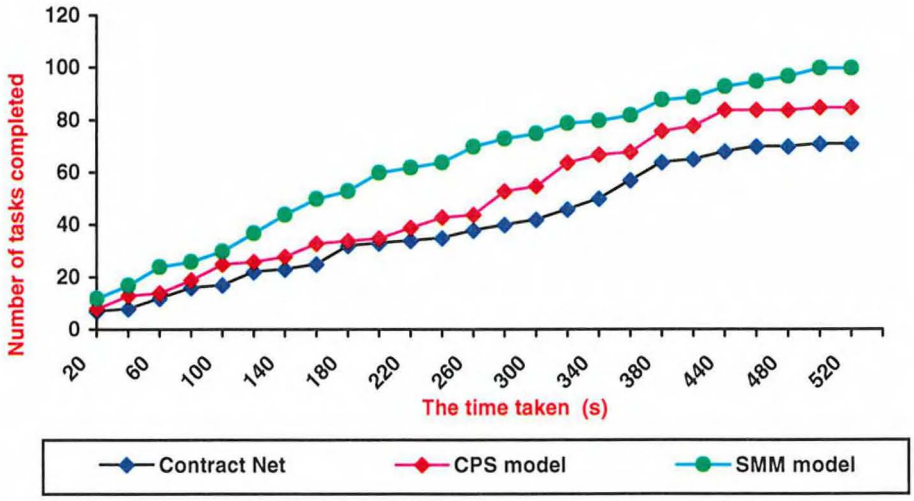


Figure 8.11 the average of the task completion rate and scope of the three systems

[6] After three time slices, if the agents repeat the same action the KS will be terminated and a new round of the task's performance is started.

Figure 8.11 also indicates the extension of the task's scope of the SMM system over the systems implementing two other models. After a certain period of time (approximately 80 seconds) the tasks completed by a system do not increase significantly. In the tests, the system implementing the Contract Net framework stoped when the completed tasks reach about 71 tasks of the 100 assigned tasks, and the CPS system stoped at about 84 tasks. This means that both models in the application domain are unable to complete all the tasks. Only the SMM system is able to complete all the tasks. What is interesting is that the CPS system failed to perform all the tasks. This is a result of the inappropriate conditions for cooperation and lack of act selection in the CPS framework. The corresponding rules in KSs under ARBS framework cannot guarantee the accomplishment of the tasks' performance. The SMM model was developed based on the implementation of the two cooperative models in the particular problem domain. The aim of the SMM model was to refine the other two models in this problem domain. The tests' results show the refinement of the SMM model.

8.3.3 The Interactions in the Cooperative Process

These sets of experiments evaluate the effect on the cooperative process in which the initial selected goals are not achieved as the agents expected. It is regarded as wasted effort. Therefore the interactions involved in these failure attempts are regarded as unnecessary or harmful interactions.

The Contract Net system was not accounted in this set of tests because the Contract Net framework does not include any uncertain attempts. Its mutual selection, based on the two-way information transfer, can ensure the efforts are not wasted. However, the lack of behaviour restriction on contractors may result in incomplete task performance under certain circumstances. This can also be regarded as wasted effort. Nevertheless, these cases have already been covered in the previous section.

Wasted effort is mainly a concern of the SMM and CPS frameworks since both include a notion of attempt. This means that both models expect some wasted efforts. It is rational to compare to what extent the efforts are wasted by the two models in the application.

To measure the wasted efforts, the number of processing cycles was used from when the agents have an *intended goal* to the time when the agents drop this goal and send it back to the panel of task_to_bid to mean that the task is incomplete. For example, the SMM system, during a task performance process the Team proposing KS (see Figure 8.3) is active at time 10 and the Team formation KS is terminated at time 12 with one of its concluding actions adding the task to task_to_bid panel. The wasted effort is 12-10, equal to 2. Similar rules are applied to the CPS system.

The tests repeat the tasks, which were generated and used to test the task completion rate in both systems. The recorded results are shown in Figure 8.12. The graph's x-axis represents the time in seconds and the y-axis represents the wasted efforts.

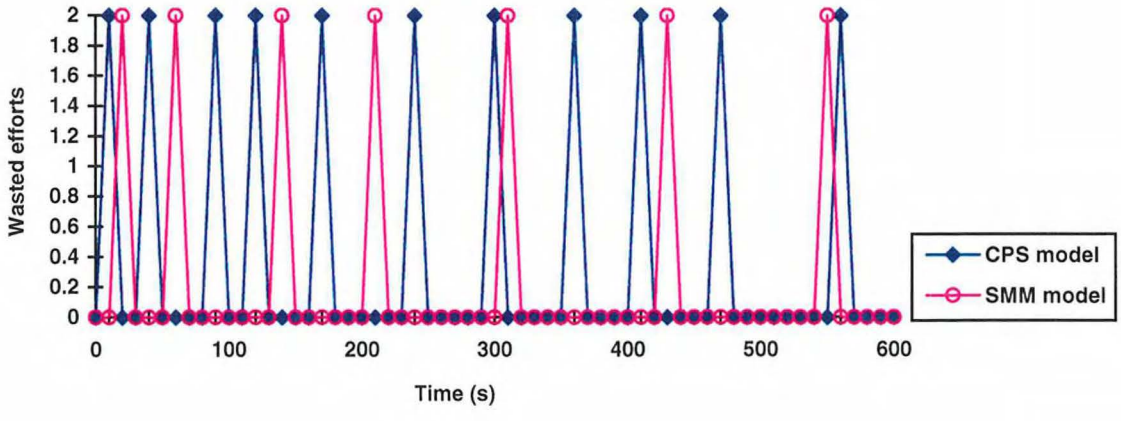


Figure 8.12 the wasted efforts in the systems

In Figure 8.12, the wasted efforts appear in a periodic pattern. This is because the wasted efforts were measured at the end of a KS when it becomes deactivated. The rules in the KS take 2 units of time. So each time when the efforts were wasted, it occupies 2

units of time. Figure 8.12 indicates that the sequence of wasted efforts appears gradually less frequent in the SMM system than in the CPS system. This shows that the SMM system has less chance of failure in its team formation attempts than its counterpart. The accumulated wasted efforts in four rounds of tests clearly indicate the decrease in the system implementing the SMM model. It is shown in Figure 8.13.

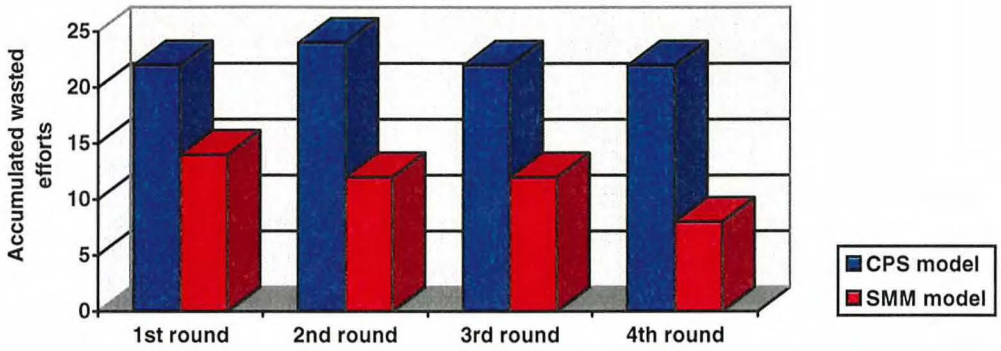


Figure 8.13 the accumulated efforts wasted in four rounds of the tests

Two reasons can be identified. One is that in the SMM model cooperation is taking place in a predicted way because of the agents' mental behaviour of 'deciding'. Another reason is that the mental adjustments carried out by agents as required in the shifting stage of the SMM model, whereby they learn from previous experience.

8.3.4 Summary

This section has reported a comparison between the systems implementing the SMM model and the Contract Nets framework and the CPS framework in the control of cooperation between multiple robots. The experiments were designed to test the claim that the SMM model is a refinement over the other two models. The tests support the hypotheses generated in the beginning of the section, which were: 1) the SMM model increases the task completion rate, 2) the SMM model increases the scope of achievable tasks, and 3) the SMM model decreases the unnecessary interactions in the task's performance process.

8.4 Discussion of Results

The aims of this empirical investigation were 1) to demonstrate the building of an application system with the logical and abstract model provided by SMM, 2) to verify the performance of the system by its different initial settings, 3) to provide an insight into the system behaviour of cooperating agents which use the SMM model.

Implementing the SMM model into an application is a relatively easy task. Any rule-based system, logic language or even conventional language is able to complete the task. To implement the SMM model into a particular application, the initial parameters defined by the SMM model, as variables in the system should be set up according to the nature of the application. That is the task loading of the system. Although the SMM model can maintain a system coherent behaviour, nevertheless, the agents' personality setting should match with tasks' weight setting since the agents' personality has also an impact on the task completion rate. However, the agents' benevolent setting achieves a high task completion rate, but it consumes more resources. It is particularly suitable for systems with a heavy task load but not for systems with scarce resources.

The experiments carried out and reported in this chapter did not aim to produce absolute evaluations of the SMM model, or the other two models. So, for example, it can not be concluded that the SMM model is better than the Contract Net or the CPS framework without specifying the domain of the problem. Rather these experiments aim to identify trends, which could expect to be observed in other instances of cooperative tasks' performance. Relating the results of the experiments back to the initial conjectures it can be seen that 1) the SMM model increases task completion rate and extends the tasks' scope, 2) in situations in which there is a high chance of joint action running into difficulties, the SMM model is able to keep the amount of wasted resources to a less degree.

As well as highlighting the benefits of the SMM model, these experiments also indicate potential drawbacks. The SMM model requests many agent's mental deliberative behaviours rather than simple reactive behaviours, it could be argued that the SMM model is inappropriate in cases where the agents are situated in a time-critical environment.

However, in the domain of cooperation between multiple robots, the agents' computation time is much smaller compared with the robots' physical movements. Typically, an atomic task takes a robot seconds to perform, and the longest agents' mental decision making is in a few milliseconds. Therefore, the SMM model is appropriate because it enforces an agent's mental behaviour rather than physical behaviour. The philosophy of the SMM model is "thinking more rather than doing more; if a mistake is taken, stop doing sooner rather than later." In other application domains, any intention of applying SMM model, the system designer should ask:

1. is unstable behaviour occasionally acceptable?

2. Is the system to be built an intelligent one rather than a reactive one?

If the answer to above two simple questions is "yes", then the SMM model could well be a good choice.

Chapter 9

Summary and Conclusions

"It is easy to say what you know but it is not easy to say what you do not know. PhD training is to build confidence to say both what you know and what you don't know."

-- Gangmin Li's diary, 26 September 1996

This chapter provides a summary of the research presented in this thesis and some conclusions. Some perceived interesting issues are listed that might be worthwhile for further research work.

9.1 Summary of the Thesis

The aim of this thesis was to develop a framework for the control of cooperation between multiple industrial robots. These robots are autonomous, pre-existing, and may continually join the system. The problem domain has been characterised as complex, open, distributed and compositional. Therefore a distributed artificial intelligence

approach was taken. Motivated by questioning the intentional theory in DAI, two dominant research approaches have been studied. Revealing their inadequacy, an organisational approach was adopted based on a fundamental belief that any multi-agent system is a form of organisation. Agents' cooperative behaviour is viewed as occurring under the joint forces of individual benefit and social benefit. A six-stage cooperation framework was proposed for multi-agent cooperation. The name of the framework, shifting matrix management (SMM), is borrowed from organisation theory to emphasise the point that cooperation is achieved by agents' shifting their positions and mental states in a matrix structure where the agents locate at different cross points that are jointly defined by the environment and their mental and physical states.

The two methods used in developing the SMM framework are classified as 'learning by doing' and 'learning by analogy'. Learning by doing means using practical experience, in this case practices which refer to actually building multi-agent systems and learning from self experience. The author built multi-agent systems based on two influential cooperative frameworks, namely the Contract Net framework and the CPS framework. Learning by analogy means drawing upon ideas from other fields that have considerable experience with their own "multiple agent" systems. The work in this thesis is based on organisation theory and economic team theory, and proposes the SMM framework for multi-agent cooperation.

To find theoretical support for the SMM framework and to provide an abstract model of SMM, a decision theory for an agent's act selection was developed. It is important because this has hitherto been neglected by DAI theorists and, without it, a complete description of the SMM model can not be produced.

The SMM framework was further developed into a general, abstract model that is based on possible worlds semantics, multiple model logic and the newly developed

quantitative decision theory. This constituted the second aim of the research which was to provide a mathematically tractable model that can cover a wider rang of applications.

Finally, to evaluate the effectiveness and the benefits of the SMM model, an empirical investigation was undertaken. A number of controlled experiments were carried out to test the hypotheses generated for each set of the experiments. The experiments were focused on three objectives.

- Investigating the effectiveness of the SMM model in multiple robots' cooperation domain. At the same time it serves the purpose of demonstrating a way of converting the abstract model into an application system.
- Varying the impact of variables in the SMM model to the overall performance of the system.
- Comparing the SMM model with other models to test the benefits claimed in this thesis.

9.2 Conclusions

The general conclusion of this thesis is the new multi-agent cooperation model. The SMM model emphasises the new viewpoint about agents' cooperation that should not only be viewed as merely accomplishment of a task performance, but also as a joint result of environmental factors and agents' physical and mental status towards these environmental factors. The environmental factors include complexity, openness, and the dynamic. The agents' mental states include motive, intention, preference and decision. The physical states include function, knowledge and current position. A cooperation model should characterise the properties of a cooperation, which is mainly dynamic, temporary and conscious. The SMM model is one attempt in this direction. Although

the model has been tested in a laboratory, these laboratory-based tests were quite simple since the robots available are simple. Much more rigorous tests are needed with different robots, in different environments. It would be useful to implement this model in other problem domains.

In developing the SMM model, a broad area of study has been encompassed. This includes robotics, DAI, organisation theory, and economic team theory. The contributions of the thesis are summarised as follows and against which the success of the thesis can be evaluated:

1. A multi-agent test-bed, based on the ARBS framework has been constructed. The implementation of the SMM model to the control of cooperation between multiple robots was performed on this test-bed.

2. An application-specific analysis of existing multi-agent cooperation frameworks has been taken based on the developed test-bed. The results of analysis provide a foundation for the new framework and later on served as a basis of evaluation of the new framework since the comparison is taken against them.

3. A novel multi-agent cooperation framework was proposed. The SMM framework regards cooperation between multiple agents as occurring in a matrix structure jointly defined by agents' motive and capability.

4. A quantitative action decision theory was developed. It reveals that an agent's mental states about an action are defined by three quantitative functions: ϕ represents an agent's belief about the chances of an event happening; ρ represents an agent's idea about the outcomes of the action determined by itself and by the environment; υ represents an agent's particular ordering of alternative outcomes of the action.

5. The SMM framework was formalised. The formalisation of the SMM framework provides a mathematically tractable model for multi-agent cooperation. It not only identifies the attitudes of agents in a cooperation process but also provides a logical relationship between these attitudes. The model can be used for both theoretical and implementation purposes.

Although the author is unable to prove the effectiveness of the SMM model in other problem domains, he is confident that the model can be implemented in other applications if the same problem characteristics exist. The system developers would need to follow the logic defined by the SMM model and configure the system with appropriate settings.

9.3 Ideas for Future Work

By the very nature of the research, it is impossible to investigate all the aspects surrounding the main problem. Actually, there are some aspects that arose as a direct consequence of this research. These could well be issues for further investigation.

The first such issue would be a further study of the impact of an agent's personality on the agent's behaviour both individually and in a social context. It was perceived in this research that the personality of agents has practical significance because of the differences in pre-existing objects that agents present. For example, in the domain of cooperation between multiple robots, robots are different in functionality, capabilities, and control mechanisms. One way of representing these robots is by agents with different personalities. It was also perceived in this research that the definition of four commonly used agents personality is based on a first-order representation. A higher order definition may need to study higher intelligent and more complex mental behaviours of agents. For example, extending a selfish agent into second-order logic

may help to understand why a selfish agent enjoys social interactions. A possible explanation is that the selfish agent realises the social interaction will benefit its self goal. Similarly, a social agent may need to be selfish in a condition which its selfish behaviour will eventually (not directly) benefit its social goal. This direction needs more study. It was also perceived in this research that there is a need to introduce the personality of agents into decision theory. In the domain of control of cooperation among multiple robots, traditional research has focused on cooperation methods based on the physical functionality and constraints of the robots. The multi-agent approach can be seen as adding a brain to these physical robots. In a social context, it is worthwhile to represent these robots with different mental states and reasoning mechanisms. The SMM model introduces a quantitative decision theory into the agent's mental behaviour to govern the agent's physical action. Further research in agent personality will no doubt play an important role in this agent's mental behaviour, and benefit to explain agents' physical behaviour.

The second issue is the need for further study of an agent's planning, particularly an agent's mental behaviour and integrating an agent's planing into multiple agents' social interactions. This research adopted a very simple "work-flow" style planning. It does not include other agents' attitudes about the plan in the planning process. Actually, human planning in a social context is considering all the participants' mental attitude. The plan produced is not merely a work process, instead it is the plan that is most likely accepted by all the possible plan participants. This means that planning needs to be expressed in a logical language which not only states the logic procedure of a goal achievement but also most possible goal achieving process and potential participants. Such a plan can be evaluated and negotiated among a group of agents in terms of benefit and cost. Therefore, an execution plan can be easily reached by a group of agents.

Works on a more fundamental nature are suggested by studies of the agents' mental states and the relationship between these mental states and physical actions. This thesis identified two basic attitudes of agents in a social context. They are motive and capability. The other attitudes are derived from these two basic attitudes. A prolonged examination is needed to investigate the effectiveness of this model to wider problem domains such as internet agents.

Finally, inspired by organisation theory, it would be worthwhile to study other methods of cooperation and coordination in natural multi-agent systems, and introduce them into artificial multi-agent systems. For example, the rewards and punishment method is an important coordination mechanism in conventional organisations. Introducing it into a multi-agent system would make agents respect organisational rules, thereafter ensuring cooperation in a more conscious and purposeful manner.

References

[Agre and Chapman 87] Agre, P., and Chapman, D., PENGI: An inplementation of a theory of activity. In *Proc. of the Sixth National Conf. on Artificial Intelligence (AAAI-87)* (1987):268-272.

[Austin 62] Austin, J. L., *How to do things with words.* Oxford University Press, Oxford, England, (1962).

[Barnard 38] Barnard, C., *The functions of the executive.* Cambridge, Mass., Harvard University Press, (1938).

[Becker 60] Becher, H. S., Notes on the concept of commitment. *American Journal of Sociol.* **66**(1960):32-40.

[Bond and Gasser 88] Bond, A. H. and Gasser, L. G., An analysis of problems and research in Distributed Artificial Intelligence. In Bond, A. H. and Gasser, L. G., (Eds.) *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers. San Mateo, CA. (1988).

[Bond and Gasser 92] Bond, A. H. and Gasser, L. G., A Subject-Indexed Bibliography Of Distributed Artificial Intelligence. *IEEE Transactions On Systems, Man, and Cybernetics,* (22) **6** (1992):1260-1281.

[Bond 90] Bond, A. H., A computational model for organization of cooperating intelligent agents. In *proceedings of Conference on Office Information Systems*, Cambridge, MA, (1990):21-30.

[Bratman 87] Bratman, M. E., *Intentions, Plans, and Practical Reason.* Harvard University Press, Cambridge, MA, (1987).

[Bratman et al. 88] Bratman, M. E., Israel, D. J., and Pollack, M. E., Plans and resource-bounded practical reasoning. *Computational Intelligence,* **4**(1988):349-355.

[**Bratman 90**] Bratman, M. E., What is intention? In Cohen, P. R,. Morgan, J. L., and Pollack, M. E., (Eds.) *Intentions in Communication,* MIT Press, Cambridge, MA. (1990):15-32.

[**Brodie and Ceri 92**] Brodie, M. L., and Ceri, S., On intelligencet and cooperative information systems: A workshop summary. *Int. J. Intell. Coop. Inf. Syst.* (2) **1**(1992):249-290.

[**Brooks 86**] Brooks, R. A., A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation,* (1) **2**(1986):14-23.

[**Brooks 91**] Brooks, R. A., Intelligence without representation. *Artificial Intelligence.* **47**(1991):139-159.

[**Cameron 94**] Cameron, S., Obstacle Avoidance and Path Planning. *Industrial Robot.* (21) **5**(1994):9-14.

[**Cammarata 83**] Cammarata, S., McArthur, D., and Steeb, R., Strategies of cooperation in distributed problem solving. *In proceedings JCAI'83,* (1983):767-770.

[**Cardarelli 94(1)**] Cardarelli, G., Palumbo, M. and Pelagagge, P. M. Critical Aspects in Multi-robot Co-operation, *Industrial Robot,* (21) **5**(1994):31-35.

[**Cardarelli 94(2)**] Cardarelli, G., Palumbo, M. and Pelagagge, P. M. Some Criteria to Help the Experimental Setup of Assembly Cells with Cooperating Robots. *3rd IEEE International Conference on Systems Integration, ICSI'94,* Sao Paulo, Brazil, August 1994.

[**Castelfranchi and Conte 96**] Castelfranchi, C., and Conte, R., Distributed Artificial Intelligence and social Science: Critical Issues, In O'Hare, G., and Jennings, N., (Eds) *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc. (1996):527-542.

[**Castelfranchi 90**] Castelfranchi, C., Social power. In Demazeau, Y. and Muller, J. P., (Eds.) *Decentralized AI,* Elsevier Science Publishers, Amsterdam, The Netherlands. (1990):49-62.

[**Castelfranchi 90(2)**] Castelfranchi, C., Miceli, M., and Cesta, A., Dependence relations among autonomous agents. In Werner, E., and Demazeau, Y., (Eds.) *Decentralized AI,* Elsevier, Amsterdam, (1992).

[**Chaib-Draa 92**] Chaib-draa, B., Moulin, B., Mandiau, R., and Millot, P., Trends in distributed artificial intelligence. *Artificial Intelligence Review*, **6**(1992):5-66.

[**Chavez 96**] Chavez, A. and Maes, P., Kasbah: An Agent Marketplace for buyingand Selling Goods. *Proceedings of the First International Conference on the Practical Application of Intelligent agents and Multi-Agent Technology (PAAM'96),* London, UK, (1996):75-90.

[**Chavez 97**] Chavez, A., Dreilinger, D., Guttman, R. and Maes, P., A Real-Life Experiment in Creating an Agent Marketplace, *Proceedings of PAAM'97,* London, UK, (1997):159-178.

[**Cohen and Levesque 90(1)**] Cohen, P. R. and Levesque, H. J., Intention is choice with commitment. *Artificial Intelligence*, **42**(1990):213-261.

[**Cohen and Levesque 90(2)**] Cohen, P. R. and Levesque, H. J., Rational interaction as the basis for communication. In Cohen, P. R., Morgan, J., and Pollack, M. E., (Eds.) *Intentions in communication,* The MIT press, Cambridge, MA. (1990):221-256.

[**Cohen and Levesque 90(3)**] Cohen, P. R. and Levesque, H. J., Persistence, intention and commitment. In Cohen, P. R., Morgan, J., and Pollack, M. E., (Eds.) *Intentions in communication,* The MIT press, Cambridge, MA. (1990):33-69.

[**Cohen and Levesque 91**] Cohen, P. R. and Levesque, H. J., Teamwork. *Tech. Rep. SRI-International,* Menlo Park, CA. (1991).

[**Conry et al 88**] Conry, S. E., Meyer, R. A. and Gasser, V. R., Multistage negotiation in distributed planning. In Bond, A. and Gasser, L. (Eds.) *Readings in Distributed Artificial Intelligence,* Morgan Kanfmann Publishers, Los Altos, CA. (1988):357-386.

[Conte et al. 91] Conte, R., Miceli, M., and Castelfranchi, C., Limits and levels of cooperation: Disentangling verious types of prosocial interaction. In Demazeau, Y., and Muller, J.-P., (Eds.) *Decentralized Artificial Intelligence*, Elsevier/North-Holland, (1991).

[Corkill and Lesser 83] Corkill, D. D. and Lesser, V. R., The use of meta-level control for coordination in a distributed problem solving network. In *proceedings IJCAI'83,* (1983):748-756.

[Coupez 89] Coupez, D., Delchambre, A. and Gaspart, P., The Scheduling of a Multiple Robots Assembly Cell. *Proceedings of the 5th CIM Europe Conference*, (1989):185-195.

[Davis and Smith 83] Davis, R. and Smith, R. G., Negotiation as a metaphor for distributed problem solving. *Artificial intelligence*, **20**(1983): 63-109.

[Decker 87] Decker, K. S., Distributed Problem-Solving Techniques: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics,* Vol SMC-17(1987):729-740.

[Demazeau 90, 91, 92] Demazeau, Y., Muller, J. P., and /or Werner, E., *Decentralised Artificial Intelligence.* 1, 2, and 3. North-Holland. (1990), (1991), and (1992).

[Dennett 87] Dennett, D. C., *The Intentional Stance.* MIT Press, Cambridge, MA. (1987).

[Durfee and and Lesser 87] Durfee, E. H., and Lesser, V. R., Using partial global plans to coordinate problem solvers. In *proceedings IJCAI'87*, (1987):875-883.

[Durfee 88] Durfee, E. H., *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, Boston, MA, (1988).

[Durfee et al. 89] Durfee, E. H., Lesser, V. R. and Corkill, D. D., Trends in cooperative distributed problem solving. *IEEE Transaction on knowledge and data engineering.* **1**(1)(1989):63-83.

[**Durkheim 83**] Durkheim, E., *The Roles of sociological Method.* Free Press, New York. (1983).

[**Engelmore and Morgan 88**] Engelmore, R., and Morgan, A. J., *Blackboard Systems.* Addison-Wesley, London, 1988.

[**Erman 80**] Ermen, L. D., Hayes-Roth, F., Lesser, V. R. And Reddy, D. R., The Hearsay-II Spaach Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Survey,* **12** (1980):213-253.

[**Etzioni 64**] Etzioni, A., *Modern organizations.* Englewood Cliffs, N. J., Prentice-Hall, Inc., (1964).

[**Feldman and Ballard 82**] Feldman, J. A. and Ballard, D. H., Connectionist models and their properties. *Cognitive Science,* (3) **6**(1982):205-254.

[**Ferber and Drogul 92**] Ferber, J., and Drogul, A. Using reactive multi-agent systems in simulation and problem solving. In Avouris, N. M., and Gasser, L. (Eds.) *Distributed artificial intelligence: Theory and praxis.* Kluwer, (1992):53-80.

[**Fikes and Nilsson 71**] Fikes, R. E., and Nilsson, N., STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* (2)**5**(1971):189-208.

[**Findler and Gao 87**] Findler, N. V., and Gao, J., Dynamic hierarchical control for distributed problem solving, *Data & Knowledge Engineering,* 2(1987):285-301.

[**Fishburn 68**] Fishburn, P. C., Uitility Theory. *Management Science,* **14**(1968):335-378.

[**Fox 81**] Fox, M., An organizational view of distributed systems. *IEEE Transactions on systems, Man, and Cybernetics, SMC-11.* (1981):70-80.

[**Gallers 89**] Galliers, J. R., *A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflict.* PhD thesis, Open University, UK., (1989).

[**Gasser 86**] Gasser, L. The integration of computing and routing work. *ACM Transactions on office information system.* (3) **4**(1986):205-225.

[**Gasser and Huhns 89**] Gasser, L. and Huhns, M. N. (Eds.) *Distributed Artificial Intelligence Volume II.* Morgan Kaufmann publishers. (1989).

[**Gasser et al. 89**] Gasser, L., Rouquette, N. F., Hill, R. W., and Lieb, J., Representing and using organizational knowledge in distributed AI systems. In Gasser, L. and Huhns, M. N. (Eds.) *Distributed Artificial Intelligence Volume II,* (1989):55-78.

[**Gasser 91**] Gasser, L., Social conceptions of knowledge and action: DAI foundations and open system semantics. *Artificial intelligence*, **47**(1991):107-138.

[**Gasser 92**] Gasser, L., DAI approaches to coordination. In Avouris, N. M., and Gasser, L., (Eds.) *Distributed Artificial Intelligence: Theory and Praxis.* Kluwer Academic Publishers, Boston. (1992):31-51.

[**Gasser 93**] Gasser, L., Social knowledge and social action: Heterogeneity in practice. In *proceedings of JCAI'93, 13th,* Chambery, France, (1993):751-757.

[**Georgeff 83**] Georgeff, M., Communication and interaction in multi-agent planning. In *proceedings of the 8th International Joint Conference on Artificial Intelligence,* Karlsruhe, Germany, (1983):125-129.

[**Gilbert 94**] Gilbert, N. and Doran, J. E., (Eds.) *Simulating societies: the computer simulation of social phenomena.* UCL press. (1994).

[**Greif 88**] Greif, I., *Computer-supported Cooperative work: A book of readings.* Morgan Kaufmann, San Mateo, CA. (1988).

[**Grohmann 96**] Grohmann, U., Paint robot in the automotive industry - process and cost optimization. *Industrial Robot,* (23) **5**(1996):11-16.

[**Hall 72**] Hall, D. T., and Schneider, B., Correlates of orhanisational identification as a function of career pattern and organisational type. *Administrative Science Quarterly,* 17(1972):340-350.

[**Hayes-Roth 85**] Hayes-Roth, B., A blackboard architecture for control. *Artificial Intelligence,* **26**(1985):251-321.

[**Hendler 91**] Hendler, J., Multiple approaches to multiple agent problem solving. In *proceedings IJCAI'91*, (1991):553-554.

[**Hewitt 85**] Hewitt, C. E., The Challenge of Open Systems, *Byte, April-1985*, (1985):223-242.

[**Hewitt 86**] Hewitt, C. E., Offices are open systems. *ACM Trans. Office Information Systems.* (3) **4**(1986):270-287.

[**Hewitt 91**] Hewitt, C. E., Open Information Systems Semantics for Distributed Artificial Intelligence. *Artificial Intelligence,* **47**(1991):79-106.

[**Hewitt and Inman 91**] Hewitt, C., and Inman, J., DAI betwixt and between: From "Intellignet Agents" to open system science. *IEEE Trans. Syst.,Man Cybernet.* SMC-21(6) (1991):1409-1418.

[**Hogg and Huberman 91**] Hogg, T., and Huberman, A., Cintrolling chaos in distribued systems. *IEEE Trans. Syst. Man Cybernet.* SMC-21(6) (1991):1325-1332.

[**Hopgood 93**] Hopgood, A. A., *Knowledge Based Systems for Engineers and Scientists*, CRC Press, Boca Raton, Florida, (1993).

[**Hopgood et al. 93**] Hopgood, A. A., Woodcock, N., Hallam, N. J. and Picton, P. D., Interpreting ultrasonic images using rules, algorithms and neural networks, *European Journal of NDT,* **2**(1993):135–149.

[**Hopgood 94**] Hopgood, A. A., Rule-based Control of a Telecommunications Network using the Blackboard Model, *Artificial Intelligence in Engineering,* **9**(1994):29–38.

[**Hopgood 97**] Hopgood, A. A., Phillips, H. J., Picton, P. D. and Braithwaite, N. St. J., Fuzzy Logic in a Blackboard System for Controlling Plasma Deposition Processes, *Artificial Intelligence in Engineering*, (1997):253-260.

[**Huhns 87**] Huhns, M. N. (Eds.) *Distributed artificial intelligence*, Morgan Kaufmann publishers, (1987).

[**Jennings and Mamdani 92**] Jennings, N. R., and Mamdani, E. H. Using joint responsibility to coordinate collaborative problem solving in dynamic environments, *Am. Assoc. Artif. Intell.,* (1992):269-275.

[**Jennings 92**] Jennings, N. R., On being responsible. In Werner, E. and Demazeau, Y., (Eds.) *Decentralised AI 3.* Elsevier Science Publishers, Amsterdam, The Netherlands. (1992):93-102.

[**Jennings 92(2)**] Jennings, N. R., *Joint Intention as a Model of Multi-Agent Cooperation.* PhD thesis, Department of Electronic Engineering, Queen Mary & Westfield College, (1992).

[**Jennings 93**] Jennings, N. R., Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering review,* (3) **8**(1993):223-250.

[**Jennings 93b**] Jennings, N. R., Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving. *Journal of Intelligent and Copperative Information System,* (3) **2**(1993):289-318.

[**Jennings and Campos 97**] Jennings, N. R., and Campos, J. R., Towards a social level characterisation of socially responsible agents. *IEE proceedings, Software Engineering*, (1) **144**(1997):11-25.

[**Koivo and Bekey 88**] Koivo, A. J. and Bekey, G. A., Report of Workshop on Coordinated Multiple Robot Manipulators: Planning, Control and Application, *IEEE Journal of Robotics and Automation,* **4**(1988):91–93.

[**Kornfeld and Hewitt 81**] Kornfeld, W. A. and Hewitt, C., The scientific community metaphor. *IEEE Transations on System, Man and Cybernetics,* (1)**11**(1981):24-33.

[**Lander 95**] Lander, S. E., Agent-based Intergration of Legacy and Reusable Software Systems. *National Science Foundation Phase I Final Report,* Blackboard Technology Group, Inc., (1995).

[**Langton 93, 94**] Langton, C. G. (Eds.), *Artificial Life, Vol1, No. 1, 2, and 3,* Fall 1993, Winter 1993, Spring 1994.

[**Lansky 87**] Lansky, A. L., A representation of parallel activity based on events, structure and causality. In Georgeff, M. P. and Lansky, A. L. (Eds.), *Reasoning about actions and plans: proceedings of the 1986 workshop,* Morgan Kanfmann Publisher, Los Altos, CA. (1987):123-159.

[**Lesser and Corkill 83**] Lesser, V. R. and Corkill, D. D., The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks,. *AI Magazine,* Fall (1983):15-33.

[**Lesser and Corkill 87**] Lesser, V. R. and Corkill, D. D., Distributed Problem Solving. In Shapiro, S. C. (Eds.) *Encyclopedia of AI,* John Wiley and Sons Publisher, (1987):245-251.

[**Lesser 91**] Lesser, V. R., A retrospective view of FA/C distributed problem solving. *IEEE Trans. Syst., Man and Cybern.* **21**(1991):1347-1363.

[**Levesque 84**] Levesque, H. J., A logic of implicit and explicit belief. In *proceedings AAAI-84*, Austin, TX, (1984)

[**Levesque 90**] Levesque, H. J., Cohen, P. R., and Nunes, J. H. T., On acting together. In *proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90),* Boston, MA., (1990):94-99.

[**Li et al. 97**] Li, G., Weller, M. J. and Hopgood, A. A., Shifting Matrix Management – a Framework for Multi-agent Cooperation, *2nd Int. Conf. on Practical Applications of Intelligent Agents and Multi-agents (PAAM97)*, London, April (1997):353-364.

[**Li et al. 98**] Li, G., Weller, M. J. and Hopgood, A. A., How to go a great good? A novel method of achieving social rationality in a multi-agent society. Submitted to *Agent and Multi-agent systems.* (1998).

[**Maes 89**] Maes, P., The dynamics of action selection. In *Proc. of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89).* (1989):991-997.

[**Maes 91**] Maes, P., The Agent Network Architecture (ANA). *SIGART Bulletin,* (4)**2**(1991):115-120.

[**Maes 94**] Maes, P., Agents that reduce work and information overload. *Communications of the ACM,* (7) **37**(1994):31-40.

[**Mead 34**] Mead, G. H., *Mind, Self, and Society.* University of Chicago Press, Chicago, IL. (1934).

[**Maines 84**] Maines, D., (Eds.) *Urban Life.* Special issue on negotiated order theory. (1984).

[**Malone 90**] Malone, T. W., Organizing information processing systems: Parallels beween human organizations and computer systems. In Zachary, W. W. and Robertson, S. P. (Eds:) *Cognition, Computation and Cooperation,* Ablex, Norwood, NJ, (1990):56-83.

[**March 65**] March, J. G., (Eds.), *The Handbook of Organizations*, Rand-McNally, Chicago, (1965).

[**Marschak and Radner 72**] Marschak, J. and Radner, R., *Economic theory of Teams.* New Haven and London, Yale University Press. (1972).

[**Martial 92**] Martial, F., *Coordinating plans of autonomous agents.* Lecture Notes in Artificial Intelligence, 610. Springer-Verlag Berlin Heidelberg, (1992).

[**McCarthy 85**] McCarthy, J., Epistemological problems of artificial intelligence. In Brachman, R., and Levesque, H. J., (Eds.), *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA, (1985).

[**McCarthy 79**] McCarthy, J., Ascribing Mental Qualities to Machines. *Tech, Rept. Memo 326, Stanford AI Lab*, Stanford, CA. (1979).

[**Miller and Drexler 88**] Miller, M. S., and Drexler, K. E., Markets and Computation: Agoric Open Systems. In Huberman, B. A. (Eds.) *The ecology of Computation*, North-Holland, Amsterdam, (1988):133-176.

[**Miller and Rice 67**] Miller, E. J., and Rice, A. K., *System of Organization*, Tavistock Publications, London, (1967).

[**Minsky 86**] Minsky, M., *The society of mind*, Simon & Schuster, New York, (1986).

[**Mintzberg 79**] Mintzberg, H., *The Structuring of Organizations*, Prentice-Hall, (1979).

[**Moore 85**] Moore, R. C., A Formal Theory of Knoeldge and Action, In: Allen, J.F. and Tate, A. (Eds), *Readings in Planning*, Morgan Kaufmann Publishers, (1985):480-519.

[**Moulin and Chaib-Draa 96**] Moulin, B. and Chaib-Draa, B., An Overview of Distributed Artificail Intelligence. In O'Hare, G. M. P. and Jennings, N. R. (Eds.), *Foundations of Distributed Artificial Intelligence.* John Wiley and Sons Publisher, ISBN 0-471-00675-0, (1996):3-56.

[**Ndumu 97**] Ndumu, D. T. and Nwana, H. S., Research and development challenges for agent-based systems. *IEE Proceedings Software Engineering*, (1) **144**(1997):2-10.

[**Newell 82**] Newell, A., The knowledge level. *Artificial Intelligence,* **18** (1982):87-127.

[**Nii 86**] Nii, H. P., Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, *AI Magazine*, **7**(1986):38–53.

[**Nilsson 80**] Nilsson, N. J., Two heads are better than one. In Davis, R. *Report on the workshop on distributed AI, Sigart Newsletter,* **43**(1980).

[**PAAM'97**] *Proceedings of the First International Conference on the Practical Application of Intelligent agents and Multi-Agent Technology (PAAM'96),* London, UK April (1997).

[**Paljug and Yun 93**] Paljug, E. And Yun, X., Experimental Results of Two Robots Arms Manipulating Large Objects. *IEEE International Conference on Robotics and Automation*, **1**(1993):517-522.

[**Papazoglou 92**] Papazoglou, M. P., Laufman, S. C. Ans Sellis, T. K., An organizational framework for cooperating intelligent information systems. *Journal of Intelligent and Cooperative Information Systems,* (1)**1**(1992):169-202.

[Parunak 91] Parunak, H. V. D. Characterizing the manufavturing scheduling problem. *J. Msnuf. Syst.* (3) **10**(1991):241-259.

[Parunak 96] Parunak, H. V. D. Applications of Distributed Artificial Intelligencein Industry. In O'hare, G. M. P. and Jennings, N. R. (Eds.), *Foundations of diteibuted artificial intelligence.* John Wiley and Sons Publisher, (1996):139-164.

[Pednault 87] Pednault, E. F. D., Formulating multi-agents dynamic world problems in the classcial planning framework. In Georgeff, M. P. and Lansky, A. L. (Eds.), *Reasoning about actions and plans: proceedings of the 1986 workshop,* Morgan Kanfmann Publisher, Los Altos, CA. (1987):47-82.

[Polllack 90] Pollack, M. E., Plans as complex mental attitudes. In Cohen, P. R., Morgan, J., and Pollack, M. E. (Eds.), *Intentions in Communication.* MIT Press, Cambridge, MA. (1990).

[Rao and Georgeff 91] Rao, A. S. and Georgeff, M. P., Modeling rational agents within a BDI architecture. In Allen, J., Fikes, R., and Sandwall, E., (Eds.), *proceedings of the second Inte. Conf. On Principles of Knowledge Representation and Reasoning,* Morgan Kaufmann, San Mateo, CA., (1991):473-484.

[Rao et al. 92] Rao, A. S., Georgeff, M. P., and Sonenberg, E. A., Social plans: A preliminary reprot. In Werner, E., and Demazeau, Y., (Eds.), *Decentralized AI 3,* Elsevier, Amsterdam, (1992):57-76.

[Rao and Georgeff 93] Rao, A. S. and Georgeff, M. P., A model-theoretic approach to the verification of situated reasoning systems. *Proc. Int. Jt. Conf. Artif. Intell., 13th,* Chambery, France, (1993):318-324.

[Reeves and Clarke 90] Reeves, S. and Clarke, M., *Logic for computer science,* John Addison-Wesley Publisher, Wokingham, England; Reading, Mass., ISBN: 0201416433, (1990).

[**Rice 76**] Rice, A. K., Individual, Group, and Intergroup Processes, in Miller, E. J. (Eds.), *Task and Organisation,* John Wiley & Sons Publisher, London, (1976):25-46.

[**Rosenschein 85**] Rosenschein, S., Formal theories of knowledge in AI and robotics. *New Generation Computing,* (1985):345-357.

[**Rosenschein and Kaelbling 86**] Rosenschein, S., and Kaelbling, L. P., The synthesis of digital machines with provable epistemic properties. In Halpern, J. Y. (Eds), *Proceedings of the 1986 conference on theoretical aspects of reasoning abut knowledge,* Morgan Kaufmann Publishers, San Mateo, CA, (1986):83-98.

[**Savage 54**] Savage, L. J., *The Foundations of Statistics.* Wiley, New York, (1954).

[**Searle 69**] Searle, J. R., *Speech Acts: an essay in the philosophy of language,* Cambridge University Press, Cambridge, Englain. (1969).

[**Seel 89**] Seel, N., *Agent Theory and Architectures.* PhD thesis, Surry University, Guildford, UK. (1989).

[**Shoham 93**] Shoham, Y., Agent-oriented Programming. *Artificial Intelligence,* **60**(1993):51-92.

[**Shoham 89**] Shoham, Y., Time for action. In *Proceedings IJCAI-89,* Detroit, MI. (1989):954-959.

[**Shoham and Tennenholtz 92**] Shoham, Y., and Tennenholtz, M., On the synthesis of useful social laws for artificial agent societies. *Proc. Am. Assoc. Artificial Intelligence.* (1992):276-281.

[**Simon 57**] Simon, H. A., *Models of man,* Wiley publisher, (1957).

[**Simon 78**] Simon, H. A., Rationality as process and as product of thought. *AER.* **68**(1978):1-16.

[**Singh 94**] Singh, M. P., *Multiagent Systems: A Theoretical Framework for Intentions, Know-how, and Communications.* Lecture Notes in Artificial Intelligence, 799. Springer-Verlag Berlin Heidelberg, (1994).

[**Sloman 97**] Sloman, *A. What sort of control system is able to have a personality?* School of Computer Science, The University of Birmingham, England. (1997).

[**Smith 80**] Smith, R. G., The contract-net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers,* (12) **C-29**(1980):1104-1113.

[**Smith 80(2)**] Smith, R. G., *A Framework for Distributed Problem Solving.* UMI Research Press, (1980).

[**Smith and Davis 78**] Smith, R. G. and Davis, R., Application of the contract net framework: Distributed sensing. *Proc. ARPA Distributed Sensor Net Symp.,* (1978):12-20.

[**Smith and Davis 81**] Smith, R. G. and Davis, R., Frameworks for cooperation in Distributed Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics,* (1)**11**(1981):61-70.

[**Sridharan 87**] Sridharan, N. S., 1986 Workshop on Distributed AI. *AI Magazine,* (Fall 1987):75-85.

[**Star 89**] Star, S. L., The structure of ill-structured solutions: Boundary objects and Heterogeneous distributed problem solving. In Gasser, L. and Huhns, M. N. (Eds.), *Distributed Artificial Intelligence Volume II,* (1989):37-54.

[**Steeb 88**] Steeb, R., Cammarata, S., Hayes-Roth, F. A., Thorndyke, P.W., and Wesson, R.B., Distributed intelligence for air fleet control. In Bond, A. H. And Gasser, L., (Eds.), *Readings in Distributed Artificial Intellgence,* (1988):90-101.

[**Steers 77**] Steers, R. M., *Orgonizational effectiveness: A Behavioural view.* Goodyear Publishing Company, Inc. Santa Monica, California. (1977).

[**Strauss 78**] Strauss, A., *Negotiations: Varieties, Processes, Contexts and Social Order.* Jossey-Bass, San Francisco. (1978).

[Stuart 87] Stuart, C., Branching regular expressions and multi-agent plans. In Georgeff, M. P. and Lansky, A. L. (Eds.), *Reasoning about actions and plans: proceedings of the 1986 workshop,* Morgan Kanfmann Publisher, Los Altos, CA. (1987):161-187.

[Tokoro 96] Tokoro, M., Agents: Towards a Society in Which Humans and Computers Cohabitate. In Perram, J. W. and Muller, J. P. (Eds.), *Distributed Software Agents and Applicaitons. Lecture Notes in Artificial Intelligence.* **1069**(1996):1-10.

[Werner 89] Werner, E., Cooperating agents: A unified theory of communication and social structure. In Gasser and Huhns (Eds.), *Distributed artificial intelligence Vol. II,* Pitman publishing, (1989):3-36.

[Wooldridge 92] Wooldridge, M., *The Logical Modelling of Computational Multi-Agent Systems.* PhD thesis, Department of Computation, UNIST, Manchester, UK., (1992).

[Wooldridge 94] Wooldradge, M., Coherent social action. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94),* Amsterdam, The Netherlands, (1994):279-283.

[Wooldridge and Jennings 94] Wooldridge, M. J., and Jennings, N. R., Formalising the cooperative problem solving process. In *proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (IWDAI-94),* Lake Quinalt, WA, July (1994):403-417.

[Wooldridge and Jennings 94(2)] Wooldridge, M. J., and Jennings, N. R., Towards a theory of cooperative problem solving. In *Proceedings of Eur. Workshop on Medell, Auton. Agents & Multi-agent Worlds (MAAMAW-94)* Odense, Denmark, (1994):15-26.

[Wooldridge 95, 96, 97] Wooldridge, M. J., and/or Jennings, N. R., Muller, J. P. Tambe, M., (Eds.) *Intelligent Agents I, II, III.* Springer-Verlag. (1995), (1996), and (1997).

[Wooldridge and Jennings 95] Wooldridge, M. J. and Jennings, N. R., Intelligent agents: Theory and Practice. *Knowledge Engineering Review.* (2)**10**(1995):115-152.

[Wooldridge] Woodridge, Mike, Agent related bibliography. http://www.doc.mmu.ac.uk/STAFF/mike/bib.html.

Appendix

A. Abstract of Selected Publications and Documented Materials

Selected aspects of the research described in this thesis, as well as related works have been documented or published elsewhere:

1. In Proceedings of 2nd int. Conf. On Practical Applicaitons of Intelligent Agents and Multi-agents (PAAM'97), London, April 1997:353-364.

Shifting Matrix Management - An Architecture for Multi-Agent Cooperation

G. Li, M. J. Weller, and A. A. Hopgood

Faculty of Technology, The Open University,
Walton Hall, Milton Keynes MK7 6AA
Tel: +44 01908 653907 Fax: +44 1908 653658
Email: {G.Li, M.J.Weller, A.A.Hopgood}@open.ac.uk

Abstract. Control of multiple robot arms with multi-agent technology is presented in this paper with the focus on cooperative behaviour required in industrial applications. A hybrid multi-layer agent architecture is proposed and its achievable behaviour discussed. The agents are organised into a team for a specific task which has a Task Manager role assigned to one of the agents. When the task is complete the team is disbanded. This architecture is termed *shifting matrix management* (SMM). With this organisation, agents are grouped by *joint intentions*, organised in a *manager-contractor* manner. The agent acts under a commitment and convention to achieve a cooperative, coherent task performance. The actual experiments suggest the rationality of the proposed architecture and scheme.

2. In Preceeding of the 3rd Young Chinese Scientists Annual Meeting, Beijing, China. September, 1998: 88-94.

Multi-Agent Systems

— Towards Computers and Humans Cohabitation

Gangmin Li

System Architecture Group, Department of Telematics,
Faculty of Technology, The Open University,
Milton Keynes, MK7 6AA, UK.

Abstract. Multi-Agent Systems is a new research field. It draws great attention from various disciplines by revealing its theoretical and practical values. This paper reviews the background, aim, motivation and context of research in Multi-Agent Systems (MAS). Comparing MAS with related research disciplines such as Distributed Artificial Intelligence, Distributed Problem solving, Open Systems, and so forth, shows what MAS really are. A particular attention is given viewing a software construction as building of a MAS. Some future research directions are suggested based on this revolutionary view point.

Key words: Distributed Artificial Intelligence (DAI), Distributed Problem Solving (DPS), Agent and Multi-Agent Systems, Software Agents, Software Engineering, Information Engineering.

3. Summited to Artificial Intellifgence in Engineering. Currently under revise.

Co-ordination and co-operation of autonomous robots through work flow control in a blackboard system

G. Li, A. A. Hopgood, and M. J. Weller

Faculty of Technology, The Open University, Milton Keynes, MK7 6AA, UK

Phone: ++44 1908 653907 Fax: ++44 1908 653658

Email: {G.Li, A.A.Hopgood, M.J.Weller}@open.ac.uk

Abstract. The co-ordination and co-operation of autonomous robot arms has been achieved using *ARBS*, an in-house blackboard system. The blackboard model of problem-solving has been adapted to accommodate the idea of blackboard partitions that are private to individual robots, as well as the usual public partitions. A work flow control scheme has been implemented by utilising *ARBS'* facility for pre-conditions on knowledge sources (i.e. on the application-specific modules). Robots bid to perform the tasks which are advertised on the blackboard. If a task is too difficult for any one robot, it is decomposed into subtasks which the robots can then bid to perform. Two tests of the system have been carried out. The first shows that, through co-operative distributed problem-solving, the system can perform a task that could not be performed by a single robot. The second test demonstrates that the system is robust to unexpected changes in the environment.

Key words. *ARBS*; autonomous robots; blackboard system; co-operative distributed problem solving (CDPS); work flow control.

4. It was accepted by PAAM'98 as a poster. It is currently under revise for Journal of Robotica.

Co-ordination and co-operation of autonomous robots by multiple charactoristic agents

G. Li, A. A. Hopgood, and M. J. Weller

Faculty of Technology, The Open University, Milton Keynes, MK7 6AA, UK

Phone: ++44 1908 653907 Fax: ++44 1908 653658

Email: {G.Li, A.A.Hopgood, M.J.Weller}@open.ac.uk

Abstract

The co-ordination and co-operation of autonomous robot arms has been achieved using characterized multi-agent technology. The blackboard model of problem-solving has been adapted to accommodate the multi-agent co-operational framework SMM. A work flow control scheme has been implemented by utilizing *ARBS'* facility for pre-conditions on knowledge sources (i.e. on the application-specific modules). Three types of tasks specified as atomic task, combination task and difficult task have been successfully performed by multiple agents with different personality. Actual experimentation have been carried out. Performing combination of different tasks shows the co-ordination and co-operation among multiple robotics arms successfully achieved by the system.

5. It was prepared for Journals of agent and multi-agent systems. It is currently under revise.

How to do a great good?
-- A novel method of achieving social rationality in a multi-agent society

G. Li, M. J. Weller, and A. A. Hopgood

Faculty of Technology, The Open University, Milton Keynes, MK7 6AA, UK
Phone: +44 1908 653907 Fax: ++44 1908 653658
Email: {G.Li, A.A.Hopgood, M.J.Weller}@open.ac.uk

Abstract. This paper describes a new principle of social rationality in a multi-agent community. The actual procedure of achieving this social rationality is also represented in a multi-agent system SMM. Where we initially narrow the multiple agents' community from an agent's society to a performance team. Then a deep analysis is given on how an individual's plan to the team task can be organized and represented. By defining the action's benefit function $B(x, a)$ and cost function $C(x, a)$ in both agent and society terms, the plan therefore can be evaluated in aspects of both doing good for the individual and the society. Following the principle of the maximum joint net benefit, The social rational plan can be generated for team to perform. The purpose of this paper is to provide a useful guidance or assistance for building a similar multi-agent system.

Appendix

B. Summary of Symbols used in the Decision Theory

X	The set of (mutually exclusive) states, x, of the environment. The uncertainty about x is expressed by a probability distribution on X.
R	The set of alternative possible outcomes, r, of an action.
A	The set of all conceivable actions, α, equivalent to the set of all functions from X to R. It is larger than the set of feasible actions, but the latter will also be denoted by A if the meaning is clear from the context.
σ	An event is, a set of states, a subset of X.
ρ	The outcome function. If the decision-maker takes action α, and the true state of the environment is x, then the outcome is $r = \rho(x, \alpha)$.
π	Probability measure on X. The probability of an event, σ, is $\pi(\sigma)$.
ϕ	Probability density function on X. For example, if X is finite, $\pi(\sigma) = \sum_{x \in \sigma} \phi(x)$.
υ	Utility function. A real-valued function on R. The utility to the decision-maker of the outcome r is $\upsilon(r)$.
ω	Payoff function. For any state x and action α, $\omega(x, \alpha) = \upsilon[\rho(x, \alpha)]$.
Ω	Expected payoff function. $\Omega(\alpha, \omega, \phi) = E\omega(x, \alpha) = \sum_x \omega(x, \alpha)\, \phi(x)$ $= \sum \upsilon[\rho(x, \alpha)]\, \phi(x)$.

C. The Syntax of the Language used in the SMM Model

1. Variables

⟨Action-var⟩ ::= α, α', α'', ..., α_1, α_2, ...

⟨Event -var⟩ ::= e, e', e'', ..., e_1, e_2, ...

⟨Agent-var⟩ ::= i, j, k, ..., i', i'',..., i_1, i_2, ...

⟨Group-var⟩ ::= g, g', g''..., g_1, g_2, ...

⟨Goal-var⟩ ::= φ, ψ, γ, φ_1, φ_2, ..., ψ_1, ψ_2, ..., γ_1, γ_2, ...

⟨Motive-var⟩ ::= ζ, ζ_1, ζ_2, ...

⟨Regular -var⟩ ::= p, q, r, p_1, p_2, ..., q_1, q_2, ..., r_1, r_2, ...

⟨Variable⟩ ::= ⟨Action-var⟩ | ⟨Event -var⟩ | ⟨Agent-var⟩ | ⟨Goal-var⟩ |

⟨Motive-var⟩ | ⟨Regular -var⟩.

2. Key Words

Event Key ::= Happened | Happening | Happens |

Mental State Key ::= Mot |

Goal | M-Goal | C-Goal |

Bel | M-Bel |

Prefer | M-Prefer |

Decide | J-Decide |

Intend | J-Intend | J-Commit |

Plan.

Action Key ::= Done | Doing | Does | Doesn't.

Attempt Key ::= Attempts | J-Attempts.

State Key ::= Know | M-Know |

Can | J-Can |

Achieves.

3. Event Expression

⟨Event-Expression⟩ ::= (⟨Event Key⟩ ⟨Event-var⟩) | (⟨Event Key⟩ ⟨Action-var⟩) |

(⟨Mental State Key⟩ ⟨Agent-var⟩ ⟨Variable⟩) |

(⟨Mental State Key⟩ ⟨Group-var⟩ ⟨Variable⟩) |

(⟨Attempt Key⟩ ⟨Agent-var⟩ ⟨Action-var⟩ ⟨Goal-var⟩) |

(⟨Attempt Key⟩ ⟨Group-var⟩ ⟨Action-var⟩ ⟨Goal-var⟩) |

◊⟨Event-Expression⟩ | ?⟨Event-Expression⟩ |

⟨Event-Expression⟩ μ ⟨Event-Expression⟩.

4. Action Expression

⟨Action-expression⟩ ::= (⟨Action Key⟩ ⟨Action-var⟩ ⟨Agent-var⟩) |

(⟨Action Key⟩ ⟨Action-var⟩ ⟨Group-var⟩) |

⟨Action-expression⟩ ; ⟨Action-expression⟩ |

⟨Action-expression⟩ | ⟨Action-expression⟩ |

⟨Action-expression⟩ ? | ⟨Action-expression⟩*.

5. Relation Expression

⟨Relation-expression⟩ ::= (Agts ⟨Action-var⟩ ⟨Agent-var⟩) |

(Agts ⟨Action-var⟩ ⟨Group-var⟩) |

(Singleton ⟨Group-var⟩ ⟨Agent-var⟩).

6. Language

⟨**Expression**⟩ ::= ⟨**Event-Expression**⟩ | ⟨**Action-Expression**⟩ | ⟨**Relation-Expression**⟩.

⟨**Language**⟩ ::= ⟨**Expression**⟩ | ¬⟨**Expression**⟩ | ⟨**Expression**⟩ ∨ ⟨**Expression**⟩ |

⟨**Expression**⟩ ∧ ⟨ **Expression**⟩ |

∃⟨**Variable**⟩⟨**Expression**⟩ | ∀⟨**Variable**⟩⟨**Expression**⟩ |

⟨**Variable**⟩ = ⟨**Variable** ⟩ | ⟨**Variable**⟩ ≥ ⟨**Variable** ⟩ | ⟨**Variable**⟩ > ⟨**Variable** ⟩ |

⟨**Variable**⟩ ⊃ ⟨**Variable** ⟩ | ⟨**Variable**⟩ ∈ ⟨**Variable** ⟩ | ⟨**Variable**⟩ ∉ ⟨**Variable** ⟩ |

A ⟨**Expression**⟩ | E ⟨**Expression**⟩ |

⟨Expression⟩ ⇒ ⟨Expression⟩.

D. Selected programs

A number of selected programs, procedures and rules in KSs are listed here as material evidence to support the claims made in the thesis.

1. Program for communication between SUN satation and PCs (POP-11 and C languages)

```
/* pc_interface.c   Gangmin Li   12/09/96 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>
#include "comm_sun_requestb.c"
#include "comm_sun_requesta.c"

int comm_sun_read(int fd, char output[128])
        {                                       /*Read from serial port a */
                int     i=0;
                int  ret_value;
                static char  index[128];
                char *p;
                p = NULL;
                if (fd < 0) {
                        puts ("port not open");
                        return(0);
                        };
                do
                {
                ioctl(fd, I_SRDOPT, RMSGD);
                ret_value = read (fd, index, 128);
                index[ret_value] = '\0';
                if (ret_value > 0)
                        {       strcpy (output, strcat(output,index));
                                p = strchr(index, '.');
        /*p points to the subtring which start with ".", if no "." appears in index p points to NULL*/
                        };}
                while (p == NULL); /* check if "." appears */
                index[0] = '\0';
                return (ret_value);
        }

int comm_sun_write(int fd, char *prompt)
        {               int     bytes;
                        /* write to serial port a */
                        bytes = write (fd, prompt, strlen(prompt));
                        return(bytes);
        }

comm_sun_request(char *input, char output[])
        {       int  fd;
                output[0] = '\0';
                fd = comm_sun_init();
```

```
                comm_sun_write(fd, input);
                comm_sun_read(fd, output);
        }

/* get_current_coord() is call comm_sun_request to find out the current coordinates of the robot and
return a pointer to 26 bytes coordinates */

char * get_current_coordb()
        {       static char *p, output[128];
                p = output;
                /*static struct _robot_coord positon;*/
                comm_sun_requestb("1.", output);
                return(p);
        }

/* move_robot(comm, coord) using communication primetives to send drive command to pc with the
destination coordinates and collect the reply signal. */

        void    move_robotb(coord)
                /*char comm[44];*/
                char coord[44];
        {       static char *p, output[128];
                p = output;
                /*strcat(coom, coord);*/
                comm_sun_requestb(coord, output);
                /*return(p);*/
        }

char * get_current_coorda()
        {       static char *p, output[128];
                p = output;
                /*static struct _robot_coord positon;*/
                comm_sun_requesta("1.", output);
                return(p);
        }

/* move_robot(comm, coord) using communication primitives to send drive command to pc with the
destination coordinates. and collect the reply signal. */

void    move_robota(coord)
                /*char comm[44];*/
                char coord[44];
        {       static char *p, output[128];
                p = output;
                /*strcat(coom, coord);*/
                comm_sun_requesta(coord, output);
                /*return(p);*/
        }

=====================================
/*  pc_interface.p  Gangmin Li, Version 2,  15/10/96 */

        uses newexternal;
        external declare pc_interface in c;
                char * get_current_coordb()
                {}
                void move_robotb(coord)
                char coord[44];
                {}
                char * get_current_coorda()
```

```
        {}
        void move_robota(coord)
        char coord[44];
        {}
endexternal;

external load pc_interface;
        'pc_interface.o'
endexternal;
```

2. Pop-11 Procedures

A number of procedures are listed here which are external to rules in a KS. It is used in a form of ARBS *runalg* to support the actions needed in rules.

```
/***** agent_ new_procedures.p      15 Oct, 1997          Gangmin Li

/*================================================================
define agent_actiona();
   vars parameters task1 parameter x y;
   performance_knowledge(task) -> parameter;  ;;; change list to word
   for x in parameter do   ;;; parameters in form of [[waist 500][..]..]
      if x matches [= ?y] then
      parameters >< ',' >< y -> parameters;
      endif;
   endfor;
   parameters >< '.' -> parameters; ;;;generate the string',500,...,0.'
   '0,1,9,2,0,0,0,0' >< parameters -> parameters;
   move_robota(parameters);   ;;; action performance  and send two reply to pannels
enddefine;

/*================================================================
define agent_actionb();
   vars parameters task task1 parameter x y;
   if agent_b_task matches [== [task for action is ?task1]==] then
    for x in parameter do   ;;; parameters in form of [[waist 500][..]..]
      if x matches [= ?y] then
      parameters >< ',' >< y -> parameters;
      endif;
   endfor;
   parameters >< '.' -> parameters; ;;;generate the string',500,...,0.'
   '0,1,9,2,0,0,0,0' >< parameters -> parameters;
   move_robotb(parameters);   ;;; action performance  and send two reply to pannels
  endif;
enddefine;

/*================================================================
define  put_current_coorda() -> list_coord;
vars x num ind elm con coord_name;
   x = conspointer_to(26);
   get_current_coorda() -> x;
   0 -> con;
   1 -> num;
   1 -> ind;
   [] -> list_coord;
  [[waist] [shoulder] [elbow] [pitch] [yaw] [roll] [gripper]] -> coord_name;
  while num < 27 do
   if  x(num) < 58 and x(num) > 47 then x(num)-48 -> elm;
            con * 10 + elm -> con;
```

```
    elseif x(num) = 44 then coord_name(ind) <> [%con%] -> coord_name(ind);
            0 -> con; ind + 1 -> ind;
    elseif x(num) = 46 then coord_name(ind) <> [%con%] -> coord_name(ind);
            coord_name -> list_coord;
    else coord_name -> list_coord;
    endif;
    num + 1 -> num;
    endwhile;
enddefine;

/*=================================================================
define  put_current_coordb() -> list_coord;
    vars x num ind elm con coord_name;
    x = conspointer_to(26);
    get_current_coordb() -> x;
    0 -> con;
    1 -> num;
    1 -> ind;
    [] -> list_coord;
   [[waist] [shoulder] [elbow] [pitch] [yaw] [roll] [gripper]] -> coord_name;
    while num < 27 do
    if  x(num) < 58 and x(num) > 47 then x(num)-48 -> elm;
              con * 10 + elm -> con;
    elseif x(num) = 44 then coord_name(ind) <> [%con%] -> coord_name(ind);
            0 -> con; ind + 1 -> ind;
    elseif x(num) = 46 then coord_name(ind) <> [%con%] -> coord_name(ind);
            coord_name -> list_coord;
    else coord_name -> list_coord;
    endif;
    num + 1 -> num;
    endwhile;
enddefine;

/*=================================================================
define check_agent_personality(paralist) -> result;
;;;paralist has two elements. One is agent name, the other is task name.
;;;paralist = [agent task]

vars  personality;
vars  social_benefit social_cost self_benefit self_cost;
;;;agent_personality=>;
/* agent_personality is a black board panel . Its contends are agent's personalities. They are:
      benevolent: agent which perform all tasks it is able to.
      selfish: best personal benefits
      social: best social benefit , does not concern individual benefit
      moderate: self + social benefit > self + social cost
 Other agent's personality can also be examined like Lazy: whatever other agent want do he never to
compete for performance.
task format: [task is [move .....]]
task weight's format is: [task ~task weight is selfbenefit selfcost socibenfit socialcost]*/

paralist(1)=>;
paralist(2)=>;

          if agent_personality matches [== [agent %paralist(1)% is ?personality] ==] and
                    task_weight matches [ == [task %paralist(2)% weight is ?social_benefit
                    ?social_cost ?self_benefit ?self_cost] ==]
           then
                    personality =>;
                    social_benefit - social_cost =>;   ;;;social_cost=>
```

```
                self_benefit - self_cost=>;  ;;;self_cost=>;
                if personality = "selfish" and (self_benefit - self_cost > 0) then
                 true -> result
                elseif personality = "social" and (social_benefit - social_cost > 0) then
                 true -> result
                elseif personality = "moderate" and ((self_benefit + social_benefit) - (self_cost +
                social_cost) > 0) then
                true -> result
                elseif personality = "benevlent" then true -> result
                else  false -> result
                endif;
        endif;
enddefine;

;;;=========================================================
define add_benefit_to_agent_gain(paralist);

        paralist(1)=>;
        paralist(2)=>;  ;;; social_benefit
        paralist(3)=>;  ;;; social_cost
        paralist(4)=>;  ;;; self_benefit
        paralist(5)=>;  ;;; self_cost
        vars social_net self_net;
        paralist(2) - paralist(3) -> social_net;
        paralist(4) - paralist(5) -> self_net;

        pr('social net:'); spr(social_net); spr('self net:');spr(self_net);

        if paralist(1) = "a" then
        agent_a_gain=>;
        agent_a_gain(1) + social_net -> agent_a_gain(1);
        agent_a_gain(2) + self_net -> agent_a_gain(2);
        agent_a_gain=>;
        elseif paralist(1) = "b" then
        agent_b_gain=>;
        agent_b_gain(1) + social_net -> agent_b_gain(1);
        agent_b_gain(2) + self_net -> agent_b_gain(2);
        agent_b_gain=>;
        elseif paralist(1) = "c" then
        agent_c_gain=>;
        agent_c_gain(1) + social_net -> agent_c_gain(1);
         agent_c_gain(2) + self_net -> agent_c_gain(2);
        agent_c_gain=>;
        else pr('mistake');
        endif;
enddefine;

;;;=================================================
define add_benefit_to_manager_gain(paralist);

        paralist(1) =>;
        paralist(2) =>;

        vars social_net self_net;
if task_manager_gain matches [== [task %paralist(2)% manager %paralist(1)% gain ?social_net
?self_net] ==] then
        pr('social net:'); spr(social_net); spr('self net:');spr(self_net);
        pr(newline); pr('agent gain previous and now:');
if paralist(1) = "a" then
  agent_a_gain=>;
```

```
   agent_a_gain(1) + social_net -> agent_a_gain(1);
   agent_a_gain(2) + self_net -> agent_a_gain(2);
   agent_a_gain=>;
elseif paralist(1) = "b" then
   agent_b_gain=>;
   agent_b_gain(1) + social_net -> agent_b_gain(1);
   agent_b_gain(2) + self_net -> agent_b_gain(2);
   agent_b_gain=>;
elseif paralist(1) = "c" then
   agent_c_gain=>;
   agent_c_gain(1) + social_net -> agent_c_gain(1);
   agent_c_gain(2) + self_net -> agent_c_gain(2);
   agent_c_gain=>;
else pr('mistake');
endif;
endif;
enddefine;

;;;======================== ==================================
define reassign_task_weight(paralist);
/*This procedure is used to reassign task's weight. Paralis  inluds two parameters: an agent & a task. 20%
of task's weight add into agent's gain, and 80% of weight equally distributed to subtasks of task.*/

paralist(1)=>;
paralist(2)=>;

vars social_benefit social_cost self_benefit self_cost;
vars social_net self_net number;
vars item subtask_list list;
0 -> number;

if  task_weight matches  [== [task %paralist(2)% weight is ?social_benefit ?social_cost ?self_benefit
?self_cost] ==]
then      (social_benefit - social_cost) * 0.2 -> social_net;  ;;; 20% belongs to manager
          (self_benefit - self_cost) * 0.2 -> self_net;

          delete ([task %paralist(2)% weight is ^social_benefit ^social_cost ^self_benefit ^self_cost],
task_weight) -> task_weight;
          aadd ([task %paralist(2)% manager %paralist(1)% gain ^social_net ^self_net],
task_manager_gain) -> task_manager_gain;
          if task_decompose_offer matches [ == [task %paralist(2)% can be decomposed into
?subtask_list] ==]
   then
      subtask_list=>;
      subtask_list -> list;
      until list = []
      do number + 1 -> number;
      tl(list) -> list;
      enduntil;
      number=>;
      (social_benefit * 0.8) / number -> social_benefit;
      (social_cost * 0.8) / number -> social_cost;
      (self_benefit * 0.8) / number -> self_benefit;
      (self_cost * 0.8) / number -> self_cost;

      task_weight=>;
      for item in subtask_list do
      aadd ( [task ^item weight is ^social_benefit ^social_cost ^self_benefit ^self_cost], task_weight) ->
task_weight;
      endfor;
```

```
   endif;
   task_weight=>;
endif;
enddefine;

/*=============================================================
define reassign_subtask_weight(paralist); */

/* The papralist have 6 parameters, task_manager, subtask, social benefit, social cost, self benefit, self cost. this function is used to change subtask's weight. Make it both social net benefit and individual net benefit becomes positive value. This comes from task manager's own gain. Therefore, it should change both subtask's weight and task manager's gain*/
paralist==>;
vars social_benefit social_cost self_benefit self_cost;
vars social_supplement self_supplement;

0 ->> social_supplement -> self_supplement;
paralist(3) -> social_benefit;
paralist(4) -> social_cost;
paralist(5) -> self_benefit;
paralist(6) -> self_cost;

if social_benefit - social_cost < 0 then
        social_cost - social_benefit + 1 -> social_supplement;
        social_cost + 1 -> social_benefit;
   endif;
if self_benefit - self_cost < 0 then
        self_cost - self_benefit + 1 -> self_supplement;
        self_cost + 1 -> self_benefit;
   endif;

aadd ( [task %paralist(2)% weight is ^social_benefit ^social_cost ^self_benefit ^self_cost], task_weight) -
> task_weight;

if paralist(1) = "a" then
   agent_a_gain(1) - social_supplement -> agent_a_gain(1);
   agent_a_gain(2) - self_supplement -> agent_a_gain(2);
elseif paralist(1) = "b" then
   agent_b_gain(1) - social_supplement -> agent_b_gain(1);
   agent_b_gain(2) - self_supplement -> agent_b_gain(2);
else
pr('task manager for task ^subtask is unknown!---- mistake');
endif;
enddefine;

/*=============================================================
Test task is difficul task or not.*/

define check_whether_task_is_difficul(list) -> result;
        list==>;
        list matches [[move == from = to =]] -> result;
enddefine;
;;;-------------------------------------------------
define check_whether_task_is_not_difficul(list) -> result;
        not( list matches [[move == from = to =]])  -> result;
enddefine;
```

```
/***** agent_test5_evaluate.p  28 Aug, 1997          Gangmin Li
================================================================
This procedure is used for agents to evaluate different plans and give each avaliabe plan an overall
score.The strategy is any steps of a plan if it is possible for the agent toperform it, it will add 10 points to
that step.
============================================================*/
/*********************************************************
;;; agentA evaluation plan: becuase that A can reach three
;;; pegs p1, p2, p3. so its evaluation is same as the rewards
;;; it can obtain from performing that step.
;;; they are:  p1 <-> p3 20 points;
;;;          p1 <-> p2 10 points; and
;;;          p2 <-> p3 10 points.
*********************************************************/
define agent_a_evaluate(plan) -> score;
          vars init final disk point;
          if not(plan = []) then
          if hd(plan) matches [move ?disk from ?init to ?final] then
          if init = "p1" then
                    if final == "p2" then 10 -> point
                    elseif final == "p3" then 20 -> point
                    else 0 -> point endif;
           elseif init = "p2" then
                    if final == "p1" then 10 -> point
                    elseif final == "p3" then 10 -> point
                    else 0 -> point endif;
           elseif init = "p3" then
                    if final == "p1" then 20 -> point
                    elseif final == "p2" then 10 -> point
                    else 0 -> point endif;
           else 0 -> point;
           endif;
   point + score -> score;
        else
        agent_a_evaluate(hd(plan)) -> score;
        endif;
        agent_a_evaluate(tl(plan)) -> score;
      else
        score  -> score;
      endif;
enddefine;

;;; *********************************************************
;;; AgentB evaluation plan: becuase that B is based on p1. It
;;; can reach p2, so it favourites step is p1 <-> p2; It may
;;; have chance to bit a part of action from p1 to p3. It is
;;; definite against any step between p2 and p3, because it can
;;; do nothing in that rang, therefore, it evaluation are:
;;;          p1 <-> p3 0 points;
;;;          p1 <-> p2 10 points; and
;;;          p2 <-> p3 -10 points.
;;;*********************************************************
define agent_b_evaluate(plan) -> score;
          vars init final disk point;
           if not(plan = []) then
          if hd(plan) matches [move ?disk from ?init to ?final] then
           if init = "p1" then
                    if final == "p2" then 10 -> point
          else 0 -> point endif;
           elseif init = "p2" then
```

```
                    if final == "p1" then 10 -> point
                    elseif final == "p3" then -10 -> point
                    else 0 -> point endif;
            elseif init = "p3" then
                    if final == "p1" then 0 -> point
                    elseif final == "p2" then -10 -> point
                    else 0 -> point endif;
            else 0 -> point;
            endif;
            point + score -> score;
          else
          agent_b_evaluate(hd(plan)) -> score;
          endif;
          agent_b_evaluate(tl(plan)) -> score;
        else
          score  -> score;
        endif;
enddefine;

;;; ***********************************************************
;;; AgentC evaluation plan: becuase that C is based on p3. It
;;; can reach p2, so it favourites step is p3 <-> p2; It may
;;; have chance to bit a part of action from p1 to p3. It is
;;; definite against any step between p2 and p1, because it can
;;; do nothing in that rang, therefore, it evaluation are:
;;;         p1 <-> p3 0 points;
;;;         p3 <-> p2 10 points; and
;;;         p2 <-> p1 -10 points.
;;;***********************************************************
define agent_c_evaluate(plan) -> score;
          vars init final disk point;
          if not(plan = []) then
           if hd(plan) matches [move ?disk from ?init to ?final] then
            if init = "p1" then
                      if final == "p2" then -10 -> point
                    else 0 -> point endif;
            elseif init = "p2" then
                      if final == "p1" then -10 -> point
                      elseif final == "p3" then 10 -> point
                      else 0 -> point endif;
                    elseif init = "p3" then
                      if final == "p1" then 0 -> point
                      elseif final == "p2" then 10 -> point
                      else 0 -> point endif;
                    else 0 -> point;
                    endif;
                    point + score -> score;
                    else
                    agent_c_evaluate(hd(plan)) -> score;
          endif;
          agent_c_evaluate(tl(plan)) -> score;
          else
          score  -> score;
          endif;
enddefine;
```

```
/*************************************************************************
  add agents evaluation result of a plan together
   add_values(value1 value2 value3) -> result
*************************************************************************/
define add_values(paralist) -> result;
        paralist(1) + paralist(2) + paralist(3) -> result;
enddefine;
/*************************************************************************
This procedure is used to count a plan's steps result is a global variable in this procedure. It has to be initilised with 0.
*************************************************************************/
define plan_step_counter(plan) -> result;
  if plan = [] then 0 + result -> result;
  elseif islist(hd(plan)) then
     plan_step_counter(hd(plan)) + plan_step_counter(tl(plan)) -> result;
  else
      1 + result -> result;
  endif;
enddefine;
/*************************************************************************
This procedure is used to find the first individual step in a plan.
*************************************************************************/
define find_step(plan) -> result;
  if atom(plan) then false -> result;
  elseif hd(plan) = "move" then plan -> result;
  else
     find_step(hd(plan)) -> result;
     unless result then
        find_step(tl(plan)) -> result
     endunless;
  endif;
enddefine;
/*************************************************************************
This procedure is used to find the individual steps in a plan.
*************************************************************************/
define find_steps(plan);
     if  not (plan = []) then
        if hd(plan) matches [move == from ?init to ?final] then
           hd(plan);
        else
        find_steps(hd(plan))
        endif;
        find_steps(tl(plan))
     endif;
enddefine;
/*************************************************************************
Change a plan into a subtask_list
********************************************************* 26 nov. 1997 ***/
define change_plan_to_subtasks(plan) -> subtask_list;
         plan(1) -> plan;
         [%find_steps(plan)%] -> subtask_list;
enddefine;
```

3. Selected KSs and Rules

Some KSs are listed here to demonstrate the format of ruls and KSs used in building a systems.

```
;;;=============================================================================
;;; initial_tasks_ks.p  An initial KS run procedure to get user's input
;;;================================================= 18/09/96 ==============
consKS (   "procedural_KS",
        [ ],
        [[notin [==] task_general]],
        [
          [runalg [task_initial[]] nil]
          [report [task entered] nil]
          [report [~agent_personality] nil]
        ],
        [ ],
        true
      ) -> knowledge_source ("initial_tasks_KS");

/*  Knoeledge source format:
     KS_type inference_mode preconditions actions ruls firablity_flag;
     KS_type: procedural_KS,  rule_based_KS, */

;;;=============================================================================
;;; agent_direct_perform_ks.p
;;; ============================================7th Oct, 1997=============

[1
        [ [in [task is ?task] task_general]]

implies
        [ [report [tasks appear on BB] nil]
        [add [task is ~task] task_to_execute]]
] -> rule ("broadcast");

[2
        [ [in [task is ?task] task_to_execute] and
                [ [in [able to perform task ~task ==] agent_a_performance_knowledge] and
                        [runalg [check_agent_personality [a ~task]] result]]]
implies
        [ [add [task for action is ~task] agent_a_task]
        [remove [task is ~task] task_to_execute]
        [report [agent a select to  perform task ~task] nil]
        [report [Check point agent personality: ~agent_personality] nil]]
] -> rule ("agent_a_select_to_perform");

[3
        [ [     [in [task is ?task] task_to_execute] and
                [in [able to perform task ~task ==] agent_b_performance_knowledge]]and
                [ [runalg [check_agent_personality [ b ~task]] result] ]]
implies
        [ [add [task for action is ~task] agent_b_task]
        [remove [task  is ~task] task_to_execute]
        [report [agent b select to perform task ~task] nil]
        [report [Check point agent personality: ~agent_personality] nil]]
] -> rule ("agent_b_select_to_perform");

[4
```

```
        [ [in [task is ?task] task_to_execute] and
                [ [notin [able to perform task ~task] agent_a_performance_knowledge] and
                        [notin [able to perform task ~task] agent_b_performance_knowledge]]
        ]
implies
         [ [add  [task for decompose is ~task] task_to_decompose]
        [report [no body knows how to perform task ~task] nil]
        [remove [task is ~task] task_to_execute]]
] -> rule ("shift_task_to_decompose");

consKS (   "rule_based_KS",
       "MI_forwardchain",
       [[in [New task is arrived] state_panel]],
       [[remove [New task is arrived] state_panel] ],
       [ broadcast
         agent_a_select_to_perform
         agent_b_select_to_perform
         shift_task_to_decompose],
       true
       ) -> knowledge_source("agents_direct_perform_KS");

;;; ================================================================
;;; agents_atomic_tasks_collection_ks.p
;;;======================================== 18 Nov, 1997=============
[10
         [ [[in [task is ?task] task_to_execute] and
        [in [task is ~task] task_general]] and
         [[in [able to perform task ~task] agent_a_performance_knowledge] or
        [in [able to perform task ~task] agent_b_performance_knowledge]]]
implies
        [ [add [task is ~task] task_failed]
        [remove [task is ~task] task_to_execute]
        [remove [task is ~task] task_general]
        [report [task ~task failed to perform] nil]
        [report [Check point agent personality: ~agent_personality] nil]]
] -> rule ("task_direct_perform_failed_collection");

[11
        [ [in [task is ?task] task_failed] and
        [in [task ~task weight is ?social_benefit ?social_cost ?self_benefit ?self_cost] task_weight]]
implies
        [
[remove [task ~task weight is ~social_benefit ~social_cost ~self_benefit ~self_cost] task_weight]
 [add [task ~task weight is ~social_benefit ~social_cost ~self_benefit ~self_cost] task_failed_weight]
        ]
] -> rule ("add_direct_perform_failed_task_weight_into_collection");

consKS (   "rule_based_KS",
       "MI_forwardchain",
       [[in [==] task_to_execute]],
       [],
       [ task_direct_perform_failed_collection
         add_direct_perform_failed_task_weight_into_collection],
       true
       ) -> knowledge_source("agents_atomic_tasks_collection_KS");

;;;============================================================
;;; agents_election_for_combination_tasks_ks.p
;;;=================================== 7th Oct, 1997 ===========
[201
```

```
        [[[in [task for decompose is ?task] task_to_decompose] and
        [notin [agent = provided decompose task ~task into ==] task_decompose_offer]]and
        [ [in [task ~task can be decomposed into ?subtask_list] agent_a_decompose_knowledge] and
        [runalg [check_agent_personality [a ~task]] result]]]
implies
        [ [add [task ~task can be decomposed into ~subtask_list] task_decompose_offer]
        [report [agent a provide that task ~task can be decomposed into ~subtask_list] nil]
        [report [agent a becomes task manager of task ~task] nil]
        [add [task manager for task ~task is a] tasks_manager]
        [remove [task for decompose is ~task] task_to_decompose]]
] -> rule ("agent_a_provide_decompose_knowledge");

[202
        [ [ [in [task for decompose is ?task] task_to_decompose] and
        [notin [agent = provided decompose task ~task into ==] task_decompose_offer]] and
        [ [in [task ~task can be decomposed into ?subtask_list] agent_b_decompose_knowledge]and
        [runalg [check_agent_personality [b ~task]] result]]]
implies
        [ [add [task ~task can be decomposed into ~subtask_list] task_decompose_offer]
                [report [agent b provide that task ~task can be decomposed into ~subtask_list] nil]
                [report [agent b becomes task manager of task ~task] nil]
                [add [task manager for task ~task is b] tasks_manager]
                [remove [task for decompose is ~task] task_to_decompose]
        ]
] -> rule ("agent_b_provide_decompose_knowledge");

[203
        [ [in [task ?task can be decomposed into ?subtask_list] task_decompose_offer]and
        [in [task manager for task ~task is ?task_manager] tasks_manager]]
implies
                [[report [task manager re-assign task weight which is 80 persent of tall weight to
                subtasks, 20 persent belongs to itself] nil]
                [runalg [reassign_task_weight[~task_manager ~task]] nil]
                 [remove [task ~task weight is = = = =] task_weight] ;;; remove the original task weight
                 [report [task weight should be removed ~task_weight] nil]
                [report [task_manager_gain ~task_manager_gain] nil]
                [remove [task ~task can be decomposed into ~subtask_list] task_decompose_offer]
                 [report [Decomposing task knowledge add to knowledge base] nil]
                [note [task ~task can be decomposed into ~subtask_list] task_decompose_knowledge]
                [runalg [add_new_list_to_task_decompose_counter[~task ~subtask_list]] nil]
                [runalg [add_new_list_to_task_to_bid [~subtask_list]] nil] ;;; splid subtask_list into
                execute format
                [report [constraints among the subtasks are generated!] nil]
                [runalg [generate_subtasks_constraints [~subtask_list]] nil]
                [report [tasks constraints: ~task_constraints] nil]
                [report [task decompose knowledge: ~task_decompose_knowledge] nil]
        ]
] -> rule ("task_manager_work_after_being_elected");

[205
        [ [in [task ?task can be decomposed into ?subtask_list] task_decompose_knowledge] and
        [in [the precedence of task ~task is task ?task_pre] task_constraints]]
implies
        [ [report [Adjust the precedence that this task follows!] nil]
         [remove [the precedence of task ~task is task ~task_pre] task_constraints]
        [runalg [adjusting_pre_constraints[~task_pre ~subtask_list]] nil]
        [report [After adjustments constraints: ~task_constraints] nil] ]
] -> rule ("task_manager_pre_constraints_adjustment");

[206
```

```
          [ [in [task ?task can be decomposed into ?subtask_list] task_decompose_knowledge]and
            [in [the precedence of task ?task_pro is task ~task] task_constraints]]
implies
          [ [report [Adjust the precedence that following this task] nil]
           [remove [the precedence of task ~task_pro is task ~task] task_constraints]
           [runalg [adjusting_pro_constraints[~task_pro ~subtask_list]] nil]
          [report [After adjustments constraints: ~task_constraints] nil]]
] -> rule ("task_manager_pro_constraints_adjustment");

consKS (   "rule_based_KS",
        "MI_forwardchain",
          [ [in [==] task_to_decompose]],
          [ [add [agents decomposing are checked] state_panel]
            [remove [agents bid is processed] state_panel]],
          [agent_a_provide_decompose_knowledge
            agent_b_provide_decompose_knowledge
            task_manager_work_after_being_elected
            task_manager_pre_constraints_adjustment
            task_manager_pro_constraints_adjustment],
        true
        ) -> knowledge_source("agents_election_for_combination_tasks_KS");

;;;=============================================================================
;;; agents_biding_and_assign_ks.p
;;;============================================== 7th Oct, 1997==============
[211
          [ [ [in [task for biding is ?task] task_to_bid]  and
           [in [able to perform task ~task ==] agent_a_performance_knowledge]] and
          [ [runalg [check_agent_personality [a ~task]] result]]]
implies
          [ [add [agent a ask for action ~task] bidings]
          [remove [task for biding is ~task] task_to_bid]
           [report [agent a ask for action ~task] nil]]
] -> rule ("agent_a_biding_for_action");

[212
          [ [ [in [task for biding is ?task] task_to_bid] and
          [in [able to perform  task ~task ==] agent_b_performance_knowledge]] and
           [ [runalg [check_agent_personality [b ~task]] result]]
implies
          [ [add [agent b ask for action ~task] bidings]
            [remove [task for biding is ~task] task_to_bid]
            [report [agent b ask for action ~task] nil]]
] -> rule ("agent_b_biding_for_action");

[213
          [[in [task for biding is ?task] task_to_bid] and
           [ [notin [able to perform task ~task] agent_a_performance_knowledge]and
           [notin [able to perform task ~task] agent_b_performance_knowledge]]]
implies
          [ [add  [task for decompose is ~task] task_to_decompose]
          [report [no body knows how to perform task ~task! futher decompose is needed!] nil]
          [remove [task for biding is ~task] task_to_bid]]
] -> rule ("agent_no_offer_biding");

[214
          [ [in [agent a ask for action ?biding] bidings]]
implies
          [ [add [task for action is ~biding] agent_a_task]
          [report [Task ~biding is assigned to agent a] nil]
```

```
        [remove [agent a ask for action ~biding] bidings]]
] -> rule ("task_manager_assign_task_to_a");

[215
        [ [in [agent b ask for action ?biding] bidings]]
implies
        [ [add [task for action is ~biding] agent_b_task]
        [report [Task ~biding is assigned to agent b] nil]
        [remove [agent b ask for action ~biding] bidings]]
] -> rule ("task_manager_assign_task_to_b");

consKS (   "rule_based_KS",
       "MI_forwardchain",
       [ [in [==] task_to_bid]],
       [ [note [agents bid is processed] state_panel]
         [remove [agents decomposing are checked] state_panel]],
       [agent_a_biding_for_action
          agent_b_biding_for_action
          agent_no_offer_biding
          task_manager_assign_task_to_a
          task_manager_assign_task_to_b],
       true
       ) -> knowledge_source("agents_biding_and_assign_KS");

;;;=========================================================================
;;; agents_action_for_subtasks_ks.p
;;;============================ 27th Oct, 1997 ============================

[216
        [[[in [task for action is ?task] agent_a_task] and
        [notin [the precedence of task ~task is task =] task_constraints]] and
    [in [task ~task weight is ?social_benefit ?social_cost ?self_benefit ?self_cost] task_weight]]
implies
        [[report [Performing task ~task by agent a] nil]
        [runalg [agent_action[a ~task]] nil];;; also add the result to agent position
        [add [task performed is ~task] agent_a_state]
        [runalg [add_benefit_to_agent_gain [a ~social_benefit ~social_cost ~self_benefit ~self_cost]]
        nil]
         [report [agent a current gian is ~agent_a_gain] nil]
        [remove [task for action is ~task] agent_a_task]
        [remove [task ~task weight is = = = =] task_weight]]
] -> rule ("agent_a_action");

[217
         [[in [task performed is ?task] agent_a_state] and
         [ [in [the precedence of task = is task ~task] task_constraints] or
        [notin [==] task_constraints]]]
implies
        [[report [Remove the constraints which takes this task as precedence!] nil]
        [add [task constraint is removed] agent_a_state]
        [remove [task performed is ~task] agent_a_state]
         [remove [the precedence of task = is task ~task] task_constraints]]
] -> rule ("agent_a_change_task_constraints");

[218
         [ [in [task performed is ?task] agent_a_state] and
         [in [task is ~task] task_general]]
implies
        [[report [task just performed is an atomic task! -- remove from general] nil]
        [remove [task performed is ~task] agent_a_state]
```

```
        [remove [task is ~task] task_general]]
] -> rule ("agent_a_delete_task_general");

[219
        [ [in [task constraint is removed] agent_a_state] and
        [in [task = subtask ~task] task_decompose_counter]]
implies
        [ [remove [task constraint is removed] agent_a_state]
        [remove [task = subtask ~task] task_decompose_counter]
        [report [task just perfromed is a subtask! -- remove from counter] nil]]
] -> rule ("agent_a_delete_task_decompose_counter");

[220
        [[ [in [task for action is ?task] agent_b_task] and
        [notin [the precedence of task ~task is task =] task_constraints]] and
    [in [task ~task weight is ?social_benefit ?social_cost ?self_benefit ?self_cost] task_weight]]
implies
        [[report [performing task ~task by agent b] nil]
        [runalg [agent_action[b ~task]] nil]
        [add [task performed is ~task] agent_b_state]
  [runalg [add_benefit_to_agent_gain [b ~social_benefit ~social_cost ~self_benefit ~self_cost]] nil]
        [report [agent b current gian is ~agent_b_gain] nil]
        [remove [task for action is ~task] agent_b_task]
        [remove [task ~task weight is = = = =] task_weight]]
] -> rule ("agent_b_action");

[221
        [ [in [task performed is ?task] agent_b_state] and
        [ [in [the precedence of task = is task ~task] task_constraints] or
        [notin [==] task_constraints]]]
implies
        [ [report [Remove the constraints which takes this task as precedence!] nil]
        [add [task constraint is removed] agent_b_state]
        [remove [task performed is ~task] agent_b_state]
        [remove [the precedence of task = is task ~task] task_constraints]]
] -> rule ("agent_b_change_task_constraints");


[222

        [ [in [task performed is ?task] agent_b_state] and
        [in [task is ~task] task_general]]
implies
        [ [report [task just performed is an atomic task! -- remove from general] nil]
        [remove [task performed is ~task] agent_b_state]
        [remove [task is ~task] task_general]]
] -> rule ("agent_b_delete_task_general");

[223
        [ [in [task constraint is removed] agent_b_state] and
        [in [task = subtask ~task] task_decompose_counter]]
implies
        [ [remove [task constraint is removed] agent_b_state]
        [remove [task = subtask ~task] task_decompose_counter]
        [report [task just perfromed is a subtask! -- remove from counter] nil]]
] -> rule ("agent_b_delete_task_decompose_counter");

[224
        [ [ [in [task ?task can be decomposed into ?subtask_list] task_decompose_knowledge] and
        [in [task manager for task ~task is ?task_manager] tasks_manager]] and
```

```
        [ [notin [task ~task subtask ==] task_decompose_counter] and
        [notin [task is ~task] task_general]]]
implies
        [ [remove [task = subtask ~task] task_decompose_counter]
         [note [task ~task can be decomposed into ~subtask_list] agent_a_decompose_knowledge]
        [note [task ~task can be decomposed into ~subtask_list] agent_b_decompose_knowledge]
        [remove [task ~task can be decomposed into ==] task_decompose_knowledge]
        [report [all subtasks of a subtask ~task have been performed successfully!! Participants learnt this
        knowledge!] nil]
         [runalg [add_benefit_to_manager_gain [~task_manager ~task]] nil]
        [report [The benefit for management are added into task manager account!] nil]
        [remove [task ~task manager ~task_manager gain = =] task_manager_gain]
        [remove [task manager for task ~task is =] tasks_manager]]
] -> rule ("agent_remove_subtask_and_learn");


[225
        [ [ [in [task ?task can be decomposed into ?subtask_list] task_decompose_knowledge] and
        [in [task manager for task ~task is ?task_manager] tasks_manager]] and
        [ [notin [task ~task subtask ==] task_decompose_counter] and
        [in [task is ~task] task_general]]]
implies
        [[remove [task is ~task] task_general]
        [note [task ~task can be decomposed into ~subtask_list] agent_a_decompose_knowledge]
        [note [task ~task can be decomposed into ~subtask_list] agent_b_decompose_knowledge]
        [remove [task ~task can be decomposed into ==] task_decompose_knowledge]
        [report [all subtasks of task ~task have been successfully performed!! All participants learnt this
        knowledge!] nil]
        [report [The benefit belongs to task manager is added to its account!] nil]
         [runalg [add_benefit_to_manager_gain [~task_manager ~task]] nil]
         [remove [task ~task manager ~task_manager gain = =] task_manager_gain]
         [remove [task manager for task ~task is =] tasks_manager]]
] -> rule ("agent_remove_generaltask_and_learn");

consKS (   "rule_based_KS",
       "SI_forwardchain",
       [[notin [==] task_to_bid] and
          [ [in [==] agent_a_task] or
             [in [==] agent_b_task]]],
       [],
       [ agent_a_action
          agent_a_change_task_constraints
          agent_a_delete_task_general
          agent_a_delete_task_decompose_counter
          agent_b_action
          agent_b_change_task_constraints
          agent_b_delete_task_general
          agent_b_delete_task_decompose_counter
          agent_remove_subtask_and_learn
          agent_remove_generaltask_and_learn],
       true
       ) -> knowledge_source("agents_action_for_subtasks_KS");

;;;=============================================================
;;; agents_shift_from_failed_to_other_ks.p
;;;=============================================================
[69
        [ [in [task for decompose is ?subtask] task_failed]]
implies
        [[report [~subtask is posed for re-decompose] nil]
```

```
        [add [task for decompose is ~subtask] task_to_decompose]
        [remove [task for decompose is ~subtask] task_failed]]
] -> rule ("shift_from_task_failed_to_task_to_decompose");

[96
        [[in [task for bid is ?subtask] task_failed]]
implies
        [ [report [~subtask is posed for re-bid] nil]
          [add [task for biding is ~subtask] task_to_bid]
          [remove [task for bid is ~subtask] task_failed]]
] -> rule ("shift_from_task_failed_to_task_to_bid");

consKS (   "rule_based_KS",
       "MI_forwardchain",
       [ [in [task for decompose is =] task_failed] or
          [in [task for bid is =] task_failed]],
       [],
         [shift_from_task_failed_to_task_to_decompose
          shift_from_task_failed_to_task_to_bid],
       true
       ) -> knowledge_source("agents_shift_from_failed_to_other_KS");
```

E. Glossary

A number of terms which carry everyday definitions e.g., motive, intention, taste, etc. and therefore may be ambiguous are defined here with respect to their use in this thesis.

Agent An autonomous and intelligent entity performing tasks on behalf of a human user. Its autonomy enables it to decide, choose and act on its own knowledge. Its intelligence prevents it from doing anything against the criteria that have been specified by the human.

Actor An actor is the executor of an agent. It performs an agent's physical behaviour. In this thesis an actor represents a robot when a robot is involved in a multi-agent system.

Agent's attitude A notion used in thesis to characterise what the world would be like if the mental states of an agent are true.

Behaviour A notion used to describe an agent's reaction to events in both the internal and external world. Two kinds of behaviours are identified in this thesis. They are mental behaviour and physical behaviour. Mental behaviour is a reaction of an agent that does not cause an action of its actor, such as a change of mental state from one to another; physical behaviour is a reaction that results in actions of the agent's actor, such as moving the forearm of a robot after the agent senses an external event has just happened.

Benevolent In this thesis a benevolent agent is an agent who performs all the tasks it has the required knowledge and ability to perform. Benevolence in this sense is the assumption that agents will perform any tasks they are asked.

Benevolent stance Benevolent stance is a conventional approach to co-operative problem solving which holds an assumption that agents are benevolent.

Belief Belief as defined in the decision theory of the thesis describes a mental state of agents that is a particular order of future alternative events. In the formalisation an agent has (and holds) a belief about a proposition p as denoted by (Bel i p) to describe that the agent has a good knowledge about p. This knowledge comes from the agent's own performance experience.

Commitment Commitment in thesis is rules, which determine the degree to which an agent persists to achieve an intention. It includes commitment both for the agent itself and for the organisation that the agent is involved and in which others agents are joint together to achieve a collective intention. In the former the intention is a self goal and in the later the intention is the organisation's goal.

Convention Convention in this thesis is embodied in rules, which determine an agent's behaviour when reconsidering an intention. Particularly, it specifies the enumerated conditions under which an agent's commitment to an object can be dropped.

Goal Goal is a particular objective an agent holds in order to fulfil its motive. In this thesis an agent having a goal with regards to its motive is a mental state where the agent is about to perform a particular task which will bring the agent a step closer to fulfilling its motive. In this thesis an agent cannot have a goal that it has no knowledge about and cannot contribute anything toward the achievement of the goal.

Intention Intention is a mental state of an individual agent whereby to fulfil its motive the agent has a persistent goal (see below). In thesis when an agent has an intention it is more than just having a persistent goal but it also has to commit itself to making a sequence of events happen, which will bring about the goal.

Intentional stance This is an approach to building systems whose behaviour can be predicated or explained by attributing to them attitudes (cognitive concepts) such as beliefs, desire, and intention.

Intentionality Intentionality is an emerging influential theoretical concept in multi-agent research, which is also called intentional theory. It is concerned with studying the degree of attributes to the attitude of intention (see above intentional stance).

Mental state The internal state of an agent. Typically the collection of beliefs, intentions, goals, etc., that characterise the agent at any one instant.

Mental behaviour This describes the mental state changes of an agent. Typically an agent updates its mental state after an external event has happened.

Modal logic The logic of necessity and possibility. The techniques of modal logic and other logics, such as temporal and dynamic logic have been used in thesis to formalise notions such as belief, goal, attempt, and so on.

Motive Motive is an initial objective an agent has when it comes into existence. It is used in this thesis to capture the basic attitude the agent holds initially.

Persistent goal Persistent goal is a mental state of agents between goal and intention. It is more than that an agent has a goal in the sense that the agent commits itself to achieve the goal to meet its motive.

Prefer Prefer represents a mental state of agents where an agent has a preference between alternative actions. If an agent prefers α to achieve its intention (see intention) on a goal then this means the agent has settled the matter on the ways of achieving the goal.

Preference Preference is used to express an agent's attitudes about choosing among available alternative actions to enable an event to happen.

Self benefit Self benefit is a measure of benefit to the individual agent. It is one of four quantitative parameters used in this thesis to evaluate tasks performed by the multi-agent system in terms of agents and system performance. The other three parameters are self cost, social benefit and social cost.

Selfish Selfish is one kind of agent personality, which is opposite to selfless. In this thesis an agent with this personality will only choose and perform tasks that bring it a positive net self benefit (see self benefit, selfless, socially responsible and benevolent).

Selfless Selfless is one kind of agent personality, which is opposite to selfish. In this thesis an agent with this personality will only choose and perform tasks that bring a positive net benefit to the society it belongs to (see selfish, socially responsible and benevolent).

Social behaviour This represents an agent's performance in a social context.

Social benefit Social benefit is a measure of benefit to the society as a whole, which is opposite to the measure of self benefit.

Social laws Social laws are the "code of conduct" in a social context. They constrain an agent's behaviour in a society where the social laws are applied. They consist of joint commitment and social convention.

Socially responsible Socially responsible is one kind of agent personality. In this thesis an agent with this personality will choose and perform tasks if their benefits for society and for agents together are greater than the total cost to society and to agents.

Taste Taste is an agent's attitude, which is concerned about choosing between alternative actions under uncertainty. Therefore, under certainty an agent's preference is the same as its taste.