AN ADAPTIVE AGENT ARCHITECTURE FOR THE DESIGN OF COMPLEX DECISION SUPPORT SYSTEM WITH APPLICATIONS TO POWER SYSTEM PROTECTION

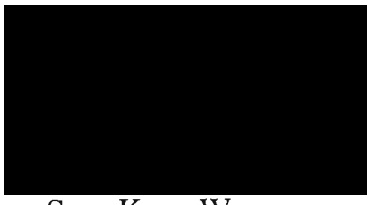
Sow Kum Wong
BA (Hons), MSc.

Thesis Submitted to the Department of Computer and Mathematical Sciences for the Research Degree of

Doctor of Philosophy
1996

Statement of Original Authorship

The work contained in this thesis has not been previously submitted for a degree or diploma at any other tertiary educational institute. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made.



Sow Kum Wong

July 1997.

Specially Dedicated To :

My dearest mother, L.T. Cheang, my family -

and

in memory of my father, C.H. Wong

Don't Give Up!!

When you feel that you are
in the midst of heading nowhere
and the whole world seems to be against you
Going down a road which does not
have any arrows of directions
to guide you to your destination
Perservere and Don't despair.

Look into one bright spot in your life
to cheer you up
as you brave on the stormy weather
Eventually, you will find that
after all the frustrated turn-arounds and tiring journeys
you've made, will worth it all
As long as you do not give up half the way
just when your patience is put to the severe test!!

Acknowledgements

I am thankful and very grateful to many people who have helped me in the course of this work, either directly or indirectly - people who provided the needed resources or technical assistance, who read and commented on the various drafts of the dissertation, who provided the much needed encouragement and moral support at all times. I list here some of these people that I would especially like to thank.

For his advice, moral support and encouragement, I wish to thank Professor Akhtar Kalam. His confidence in me gave me the strength to push on, helped to keep
my research on track and to complete this thesis on time.

Professor Clement Leung for his supervision, valuable advice and comments. His suggestions and recommendations regarding the final draft of the thesis were very much appreciated and I thank him for it.

I would like to thank Charles Biasizzo and Adam Klebanowski for their expert advice and patience in explaining the terms and concepts of power system protection to me.

The former and the current Head of Department of Computing and Mathematical Science - Associate Professor Ian Murray and Associate Professor Neil Barnett for granting me the scholarship, providing me the opportunities to attend some conferences and giving me the financial support through sessional teaching.

The Head of Department of Electrical and Electronics Engineering, Associate Professor Wally Evans for kindly accepting me into the department, accomodating and providing me with the necessary facilities.

The technical staff, Rajendran Ponnusamy, Damon Burgess and Zoltan Varga, for their kind assistance and for tolerating my endless problems with the computers and printers especially.

My constant source of encouragement, my mentor, my guardian - Ted Alwast, I am indebted and forever grateful to him. His great confidence in me gave me the much needed inspiration and encouragement to pursue on to my Master and then my Doctorate. His faith in me gave me the strength that I need to carry on especially at times when I get weary and just about to give up. He is always there to guide and advise me, and give me the moral support - in my work and in my personal matters. Without him, I would not be where I am today. My deepest and sincere thanks to him.

To Robert Hinterding, Iwona Miliszewska, Grace Tan, the administrative staff of Computing and Mathematical Sciences Department and all my other friends and colleagues. Their kind advice, generous assistance and for being there for me meant a lot and it made my stay at Victoria University a very pleasant, enjoyable and memorable one.

My family, of course, especially my mother for her love, understanding, encouragement and being my source of inspiration. My second sister and my youngest brother who are here with me, for their patience, care, understanding and encouragement. To the rest of my other brothers and sisters, even though they are not here with me, they have always been my source of encouragement and inspiration. Without my family, I would not be here at all.

May GOD Bless Them All.

Preface

In the early 1990s, there was a notable lack of intelligent systems in the area of power system protection. Moreover, there was no legacy software that would design a coordinated protection scheme for a power system, although there was considerable interest in such applications shown by people from the industry. However, to build a good, robust and efficient application system, the architecture must be designed properly and with care. It is the architecture that eventually builds up the core of an operational system. Therefore, serious thought and consideration must be given to the study of the architecture. Unfortunately, literature review showed that not many research projects were undertaken in this area. This prompted investigation into the analysis and design of a generic agent architecture for the development of an intelligent system which could be applied to any application domain.

The approach used here to design the agent architecture is an adaptive and incremental approach. One of the main attractive features of the agent architecture is the dynamic knowledge component of the system. The knowledge base can grow and modify itself automatically so that it can adapt itself to the ever changing environment and demands. The design of the agent architecture is developed and built using a layered architecture approach. Layering is a very powerful technique that enables effective representation and integration of various techniques and paradigms into one single architecture. Communication is performed via message passing between the adjacent layers.

One of the advantange of this adaptive approach is the development of intelligent agents which imitate the behaviour of human experts during problem

solving. The agents attempt to apply their past experiences or skills when resolving a problem. Over the time, as they go through more problem solving tasks, their experiences, skills and knowledge also grow. The technique used to selectively retains the experience, copes with the memory requirements for knowledge expansion, while the organisation of the memory layer allows the agent to confidently add the reliability factor to the proposed solution.

There are basically three main types of agents developed for the implementation of an intelligent multiagent system in this thesis. These are the Interface agent, Coordinator agent and Design agent.

The agents can be organised in a number of ways to construct different system architectures. Architectures can differ in many ways including the organisation of the architecture - hierarchical or heterarchical, the nature of the control mechanism - distributed or centralised, and the number of entities or processing units employed in the system. There are a number of factors that contribute to the different architectures such as the types of agents employed, the requirements and environmental issues. However, the degree of agent's autonomy is probably the main influencing factor. In this thesis, different types of agents of varying degree of autonomy are employed and organised in different ways to develop a number of system architectures. This is possible because like many design problems, there is no one absolute design solution for a system architecture. Causal links are studied and the agent's existence is justified as each is added to the system architecture.

The different system architectures developed represent multiagent systems which utilise new technologies and paradigms including agent technology,

distributed problem solving, multiparadigm approach and employment of multireasoning strategies. In addition, a blackboard system and a federated system organisation are also employed.

The reasoning paradigms employed are case based reasoning, rules, explanation based and argumentation. However, case based reasoning appears to be the most natural reasoning technique that imitates the human approach to solve design problems. Therefore, case based reasoning is chosen as the main reasoner in the system.

The three system architectures have been produced by utilising different organisation of the agents developed in this thesis. It should be pointed out that the autonomy of the agents decreases with each system listed below :

- System architecture using distributed knowledge in a knowledge based environment;
- System architecture using distributed knowledge/shared knowledge in an object oriented environment;
- System architecture using distributed knowledge within a federated system framework in an object oriented environment.

The first system architecture consists of three types of semi-autonomous agents namely Interface agent, Coordinator agent and Design agents which are organised hierarchically. The hierarchical organisation supports the natural grouping of functionally related agents to facilitate cooperative problem solving. Communication between the agents is accomplished using message passing paradigm. The Interface agent is responsible for all interactions between

the system and the outside world. The Coordinator agent is responsible for decomposing a problem into smaller problems and also for coordinating the partial solutions into a coherent and integrated solution to the original problem. The Design agents also known as problem solving agents are specialised agents. Each specialises in a particular aspect of the application domain. They apply their respective expertise to generate partial solutions to the problem.

The second system architecture is similar to the first architecture with the addition of a shared knowledge base and a blackboard system. The common knowledge possessed by the various agents in the system is shared and communication is carried out via the blackboard. The partial solutions proposed by the Design agents are posted to the blackboard, also known as the Global Database. The Coordinator agent then prepares the final solution using the information posted to the blackboard.

The third system architecture is similar to the second architecture where a Global Base is employed to house a community agents. However, in this federated system, the agents surrender their autonomy to the Facilitators. The Facilitators are responsible in selecting and placing partial solutions, represented by the agents, in the Global Base. The Controller/Modifier agent then organises and modifies the agents in the Global Base so to produce the final design solution.

The system architectures have been implemented in order to study the feasibility and advantages/disadvantages of each architecture more accurately.

System evalutions are carried out based on the three system architectures produced. Each system architecture is studied, compared and analysed in order to deduce the best solution for the design of the architecture given all the constraints including operating environments.

The thesis closes with a final conclusion regarding the work accomplished in the thesis. A number of recommendations regarding possible enhancements to the system for future investigations are also presented.

List of Abbreviations

ABSI	Agent Based Software Interoperation
ACL	Agent Communication Language
AEPD	Automated Electric Plat Design
AI	Artificial Intelligence
API	Application Program Interface
BA	Bus Agent
cts	current transformers
CA	Controller Agent
CAD	Computer-Aided Design
CAPE	Computer-Aided Protection Engineering
CBR	Case Based Reasoning
CIDIM	Cooperating Intelligent systems for Distribution Management systems
CPU	Central Processing Unit
DAI	Distributed Artificial Intelligence
DARES	Distributed Automated Reasoning System
DPS	Distributed Problem Solving
DSS	Decision Support System
EMMA	Enterprise Modelling and Management System

FA/C	Functionally Accurate / Cooperate
FMS	Flexible, Programmable, Integrated Automation
GDSS	Group Decision Support Systems
GIS	Geographic Information System
GRATE	Generic Rules and Agent Model Testbed Environment
IA	Interface Agent
IDSS	Intelligent Decision Support System
ISPP	Intelligent System for Power Protection
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
LA	Line Agent
NASA	National Aeronautics and Space Administration
NEST	Networked Expert Systems Testbed
OODB	Object Oriented Databases
OODBMS	Object Oriented Database Management System
PAS	Persistent Application Systems
P/T	Petri Net
SIF	Simplified Interchange Format
URD	Underground Residential Distribution

List of Figures

List of Tables

Table of Contents

PART II
The Work:
The Architecture Design of a Generic Agent

PART III
The Solution:
Application and Implementation of Different System Architectures

PART IV
The Conclusion:
Discussion and Future Work

Appendices

Addendum

List of Refereed Publications:

1. Development of a Decision Support System for Power System Protection System Design, Analysis and Assessment,
   The 29th Universities Power Engineering Conference, 1994 in University Galway, Ireland, Proceedings of UPEC '94, pp.39-42.
   S.K Wong, A. Kalam

2. A Decision Support System Using Case Based Reasoning in Power System Protection,
   Australiasian Universities of Power Engineering Conference in University of South Australia, Australia, Proceedings of AUPEC '94, pp.682-687.
   S.K Wong, A. Kalam, A.Klebanowski.

3. Incorporating Distributed Problem Solving Technique into a Case Based System,
   Eighth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems 1995 in Melbourne University, Australia, Proceedings of IEA/AIE '95.
   S.K Wong, A. Kalam, C.H.C Leung.

4. Development of A Power Protection System using an Agent Based Architecture.
   International Conference on Energy Management and Power Delivery in Singapore, Proceedings of EMPD 95, pp. 433-438.
   S.K Wong, A. Kalam.

5. Distributed Intelligent Power System Protection using Case Based and Object Oriented Paradigms.
   International Conference on Intelligent Systems Applications to Power Systems in Orlando, Florida, USA, Proceedings of ISAP '96, pp.74-78.
   S.K Wong, A. Kalam.

6. Intelligent Power System Protection Using Agent Technology.
   The 31st Universities Power Engineering Conference (UPEC 96) to be held in Technological Educational Institute Iraklio, Iraklio, Greece, 18-20 September 1996, pp.858-861.
   S.K Wong, A. Kalam.

7. Agent Based Technology and its Application in Power System Protection. *Accepted for publication* in the International Journal of Power and Energy Systems vol.22 no.17, The International Association of Science and Technology for Development (IASTED), Canada.
S.K Wong, T. Alwast, C. Biasizzo, A. Kalam.

8. An Agent Approach to Designing Protection Systems.
Published in Sixth International Conference on Developments in Power System Protection (DPSP 97) to be held in University of Nottingham, UK, 25-27 Mar 1997.
S.K Wong, A. Kalam.

PART I

THE PROBLEM

The Intelligent Multiagent System and Design Problems

Chapter One

INTRODUCTION

Abstract

Design of a system architecture is like an art, as there is no one absolute design for a system. There are many issues to be resolved, for example, performance requirements, integrity, reliability, system requirements and expectations. There are various designs that could be applied to build a system. However, that does not necessarily means that the architecture is appropriate at all. The main focus of this research is on the study of architecture and the presentation of a new approach to the design of architecture. This architecture is then applied to the development and implementation of an intelligent system. The approach - is an incremental and adaptive process where components are developed and their existence are justified and assessed with the environment and task requirements before they are integrated and put together to construct the complete architecture. This chapter gives an introduction to existing systems, current technologies and future trends. It also presents the research objectives, and the contributions of the research.

1.0 Introduction

The design of a system architecture is more like an art. There is no absolute design for one system. There are a variety of ways in which the architecture of a system could be designed and numerous variations of techniques and approaches that could be applied and integrated. There may not be much difference in the system, once implemented, from the user point of view. However, there could be a great diversity in the designing stage to the

development and implementation of the same system based on the different architectures.

The design and the eventual implementation of a system involves a number of issues which need to be resolved. Among these issues are:

- inter-process communication;
- module structure;
- software and hardware constraints;
- reliability;
- integrity;
- performance requirements;
- application requirements.

In distributed systems, they are further characterised by either :-

- physical separation of computing resources;
- logical separation of multiple processing units.

In most systems, and more so in the distributed systems, some sort of communication between the resources or the processors is required. In the latter systems, the processing units may be housed together or be physically separate and may or may not share memory [Vickers and McDermid '93].

Furthermore, when designing a system architecture, one of the most important issue that needs to be resolved is knowledge representation. However, in the design of a traditional expert system, there may not be much issues to

consider and the designer usually do not have to worry about the knowledge representation. The traditional expert systems usually employ rules in the knowledge base and relatively, they have one inference engine only.

Most of the application systems developed even up to the past few years, utilised the more conventional approach of building one single large expert system and employing production rules to perform the inferencing. These systems can easily exceed to hundreds and thousands of formal expert rules in the knowledge base. Rule based systems are simple and easy to implement, however one of the main drawbacks of such systems is the problem of maintenance. A change in one rule or a change in the environment can have a big impact and undesirable effects on the other rules in the knowledge base. Other limitations of rule based systems include :

(i) brittleness of the system;
(ii) the domain is restricted to a narrow and well-defined area.

Hence to build a robust, flexible, maintainable and expandable system, a system cannot depend on just production rules alone. Therefore, other methodologies have been introduced and used to either replace or integrate with rules. Some of these methodologies for example, are memory based reasoning, case based reasoning, explanation based reasoning, reasoning under uncertainties, causal reasoning, function based reasoning, experiential reasoning, goal based reasoning and so on. In fact, many application systems nowadays are hybrid systems that employ multiparadigm reasoning techniques. Different reasonings and techniques are needed to complement one another and to make up the limitations of the other methodology.

In addition to adopting multiparadigm approach and using AI techniques, distributed problem solving technique and agent based architectures seem to be the present and future interest and direction for developing intelligent systems. The current growth of interest is more widespread in areas such as medical, concurrent engineering, diagnosis, manufacturing, networking, aerospace and robotics [Buttler 95, Hayes-Roth 94, Huang *et al.* 94, Jennings *et al.* 93, Khedro *et al.* 93, Peligry 94, Rossomando 92, Smith and Slade 92, Winter *et al.* 95, Wittig *et al.* 94]. Distributed problem solving employs knowledge sources, also known as agents or expert systems. This technique not only allows knowledge bases to be represented in multiple forms and also to reside in different environments/platforms.

However, regardless whether the agents and/or knowledge are distributed within the same or different platforms, some sort of coordination or cooperation between the agents would be necessary. Cooperation in these systems is important, and could mean interactions and communication between agents directly or indirectly so that a coherent and consistent solution can be constructed.

1.1 Existing Systems, Current Technologies and Future Trends Especially in Power Systems

While the main thrust of the research is to develop an adaptive intelligent system which could be applied to any domain, it is imperative, nevertheless, to select an application area where the viability of the architecture of the system could be thoroughly demonstrated. Moreover, the application domain should

not only offer a challenging problem but which could also benefit from the new technology.

The application domain chosen for the implementation of the intelligent multiagent system is power system protection. Power system protection is an interesting area which represents a class of design problems where protection schemes can be designed for parts of or the whole power system. Even though different schemes can be applied to a power system, it is imperative to ensure that the schemes are reliable. Reliability is an important factor because of the potential risk and dangers involved when a protection scheme fails to operate when a fault occurs in a power system.

Most of the older generation systems built in power protection or in any other areas, for that matter, employ one relatively simple inference engine to work on the knowledge base. The knowledge base is maintained in a particular representational format, usually as production rules. As reported by Wielinga *et al.* [Wielinga *et al.* 92], the knowledge base of the more traditional systems, eg. MYCIN, hides various important properties of the reasoning process and of the structure of the knowledge in the application domain. Certain rules, or parts of rules, remain implicit in such knowledge based systems and this implicitness impairs the acquisition and refinement of knowledge, reuse of the system, its explanatory power and the assessment of its relation with other systems.

Recent attempts to develop larger and more complex knowledge based systems have revealed the shortcomings and problems of centralised, single expert system architectures [Adler *et al.* 92]. As a result, there are now an increasing number of artificial intelligence (AI) systems being developed,

diverting from the more traditional and monolithic systems. According to Decker [Decker 87], one of the most powerful aspects of AI approach to problem solving is the ability to deal with uncertain and incomplete information. Consequently, there is a fast growing interest in the use of multi agent paradigm in homogeneous as well as in heterogeneous systems. As Adler *et al.* in [Adler *et al.* 92] said :

> *"Multi agent refers to the coordination of a number of agents (goals, knowledge, plans and so on) to solve problems, that is, an emergent system where the whole is greater than the parts."*

In the recent conferences on power systems [EMPD 95, IEA/AIE 95, ISAP 96], it is revealed that there is a significant inflation and a strong emerging trend in the use of AI techniques in the development of intelligent systems in various areas of power system. Such AI techniques include fuzzy logic, genetic algorithm, case based reasoning, temporal reasoning.

Genetic algorithms (GAs) has been a popular topic of research in the past few years. It was developed in the mid 1960s by Holland [Holland 92]. Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics, where the principle of '*survival of the fittest*' is applied [Goldberg 89, Jenkins 92,]. It is a stochastic, iterative, evolutionary, general-purpose search strategy which is based on the principles of population genetics and natural selection [Galletly 92]. It has been applied in many areas especially where optimisation is the main issue of interest. Celko [Celko 93] has applied GA in databases to find the optimal set of indexes based on a set of queries while Hinterding and Juliff [Hinterding and Juliff 93] applied GA for stock

cutting in order to minimise wastage. In power system, GAs have been employed in the area of distribution [Brown *et al.* 96, Choi *et al.* 95], security assessment, control and planning [El-Sharkawi and Huang 96, Lai and Ma 96, Oyama 96] and also in a real time system [Sakata and Iwamoto 96].

Fuzzy logic is another AI technique that has been popularised over the recent years. It has been widely applied to areas where uncertainties persist, especially in the areas of robotics and control [Kagan and Oliveira 96, Stoica and Wingate 93, Li *et al.* 95, Isomursu *et al.* 95, Hill and Schmalenbach 95]. The National Aeronautics and Space Administration (NASA) is another centre where researchers are using fuzzy logic for a number of applications, from controlling the docking of spacecraft, to controlling ambient conditions within space habitats [Hoffmann 94]. Fuzzy logic has also become an active research topic in power system. It has been applied in the areas from fault diagnosis and power system diagnosis [Chang *et al.* 96, da Silva and Zebulum 96] to controllers, outage determination and stabilisers [Dash and Panda 95, Huang and El-Sharkawi 96, Hariri and Malik 96, Park *et al.* 96, Sumic and Vidyanand 96].

Interesting enough, case based reasoning represents a general paradigm for reasoning from experience [Slade 91]. It is a natural approach which simulates the human's approach to problem solving, has somehow escaped the attention of researchers in power system[1]. The number of research in the application of case based reasoning to develop case based systems has increased many folds the past few years [Case-Based Reasoning 91, EWCBR 93, AIC 93, AAAI 94]. Its

[1] Case based reasoning is used as the main reasoner in developing the system in this thesis. Refer to Appendix F for more details on case based reasoning paradigm.

area of applications has grown which include those using cases to resolve disputes, design, planning, legal reasoner and diagnosis [Koton 88, Sycara 88, Kolodner 91, Kolodner 93, Bardasz and Zeid 93].

Expert systems has used to dominate the development of application programs in power systems [Lo and Nashid 93]. Now, there is a growing number of knowledge base and decision support systems. For example, McArthur *et al.* [McArthur *et al.* 96] developed a knowledge based decision support system for protection engineers. Other knowledge based systems have been developed in the areas of fault location and diagnosis [Eickhoff *et al.* 91, Marin and Banerjee 94, Tang *et al.* 96, Vazquez *et al.* 96], power system restoration and power plant upgrading [Jiang and Teo 95, Tangen and Stoa 96] and real-time plant protection and safety [Taylor *et al.* 90].

Over the recent years, there has been a steady growth in the use of object oriented paradigm and object oriented database management systems to develop systems in many diversified areas. In power system, object oriented database has been used in distribution network and protection areas [Kawamura and Wakizono 96, Wong *et al.* 95, Chen *et al.* 94] has applied object oriented paradigm to develop an expert system for power system on-line restoration.

More interestingly, there is a new wave of interest in the development of intelligent systems in power system. Even though the growth of the interest in power system is slow, it is definitely growing. For example, some of the intelligent systems have been built in the area of power distribution [Hagg and Ygge 95, Verho *et al.* 95], load forecasting [Parlos *et al.* 96], power control [Lee

et al. 94, Elders 96, Lee *et al.* 96] and power protection [Wong *et al.* 95, Wong and Kalam 96, Wong *et al.* 96].

Looking at the direction in which the research is going, there is a strong indication that this healthy but forceful trend will definitely continue on well into the future and into the 21st century. This statement is substantiated by the study made by Holen [Holen 96] on the future need for intelligent systems in the changing utility environment. Holen concluded that the new environment is more complex from the market, business, demand and regulation views, and that everything is undergoing a rapid change. This resulted in higher stress on people and systems and thus, giving rise to the need for intelligent systems to accommodate the changing world.

1.2 Problem Statement

Literature survey revealed that there has been a very small number of research projects in the development of intelligent systems in power system protection. Previous systems developed in this domain employed traditional approaches and have relied on rules to represent the domain knowledge. One of the major limitation or drawback of using the traditional approach has been that such knowledge is not amenable to automatic adaptation to new contexts. Modifying or updating the knowledge base is known to be a tedious task which can be very difficult, if not impossible in some systems.

For example, the two systems developed by Kalam *et al.* and Liu *et al.*, [Kalam *et al.* 91, Liu *et al.* 94] relied on rules. Both systems are relatively small

systems which focus on the design of protection scheme for transformer components only. Further literature survey reveals that there is no existing system in the design and assessment of a power system which utilises agent technology and multiparadigm techniques.

However, the construction of the intelligent system itself is no simple task too. They have to undergo the development life cycle which includes study of problem domain, analysis requirements, design, implementation, testing and maintenance. Every phase is important and should be carried out with due considerations to ensure the success of the system in the future. The design of the architecture should be emphasised. Once the analysis requirements are completed, it is very important that the architecture is designed with care and evaluations should be done on the system components and other alternative design. The efficiency, robustness and adaptability of the system to changes or expansion would depend on the architecture of the system and how it was constructed.

Even though, there is an awareness on the importance of the architecture of a system, there is still a lack of significance and emphasis in the study of architecture. However, there are currently some approaches or methodologies developed to the design of architecture. The solution of design problems necessitates considerable intelligence, flexibility, and adaptability. A versatile architecture is necessary to support such a task. In this thesis, a new adaptive approach for the design of a generic architecture is given. The architecture is constructed to enable the agent to behave and operate more like an expert. This is a novel approach which constitutes a major part of the thesis. Investigations are further carried out to determine how agent's autonomy and organisation of

agents can affect the system architecture. The architectures are then applied to develop an adaptive intelligent multiagent system in power system protection - that is, to design protection schemes for a power system or parts of it. The application of multiagent system is new and has never been applied to the area of power system protection. As a conclusion to the study, different system architectures were designed, prototyped and compared where evaluations were then drawn.

1.2.1 Nature of the Research

The aim of this research project is to apply artificial intelligence techniques to the development of a new generic and adaptive intelligent agent for the development of an intelligent multiagent system in any disciplinary area. Being a generic agent, its architecture can be modified and adapted easily to construct different types of agents to meet the requirements of different systems. In order to build an agent which is adaptive and intelligent, the agent architecture is constructed using the analogy of the human mind. The advantages of using this approach is that the agents developed simulate the human natural approach to problem solving and it possess the similar skills and capability of a human expert in decision making and solving problems in the presence of uncertainties and limited knowledge.

An intelligent system is expected to behave intelligently in a consistent and reliable manner and to mimic the behaviour of human experts of the same domain. The application domain in which the intelligent system is applied to in this research is power system protection. The objective of the intelligent system

is to assist the engineers in the design of protection schemes for power system or a part thereof. Designing protection schemes requires more than just theoretical knowledge obtained from references or textbooks, experience is very important as the decisions made during problem solving are subjective and follow the 'rules of thumb'. However, it can be a repetitive task as well, when the engineers are involved in designing protection schemes for the same or similar components of a power system more often than not. Thus, the need arises for an intelligent system that would assist the engineers in the design and assessment of power system protection scheme.

The system would provide assistance and facilitates in decision making, and in expediting repetitive work so that the engineers could spend more time in other 'qualitative' work. Besides, this system could also serve as a library of references where detailed information and knowledge about the protective schemes and relays used could be assessed easily that is, at fingertips (so to say), rather than, being distributed in various sources and locations. Moreover, new inexperienced engineers could learn, and acquire knowledge and experience from this type of systems more effectively. But nevertheless, such system or any other intelligent or decision support systems for this matter, will not be able to replace human experts.

As Burstein *et al.* puts it [Burstein *et al.* 94] :

> *" An important aspect of decision support system has been the philosophy that the system exists to support the decision maker, not to make the decisions for him. "*

In other words, an intelligent or decision support system is meant to provide assistance with problem solving techniques and/or with additional knowledge about the problem domain itself rather than giving the prescription of outcomes.

In order to implement the above system, the architecture has to be designed and the decisions on the use of appropriate artificial intelligence (AI) techniques must be made. While there are many studies and research carried out on the different AI techniques and the development of intelligent systems using the various methodologies and paradigms, the same cannot be said on systems' architectures. Until recently, there were no serious or formal studies on systems architectures. The Knowledge Systems Lab at Stanford University, US, [Hayes-Roth 94, Hayes-Roth *et al.* 94] and The Cognition and Affect Group at University of Birmingham, UK, [Read 93, Read 94, Read and Sloman 93, Sloman 92, Sloman *et. al* 94, Sloman 95] are ones of the very few minority who pursued into the study of architecture from a formal viewpoint. Sloman [Sloman 95] came up with a new methodology in the study of architecture - design based approach which explores an architecture of intelligent system as a possible design solution to a collection of requirements derived in part from analysing aspects of human mental life. Read [Read 93, Read and Sloman 93, Read 94] extended the design based approach to provide a methodology for studying 'emotion' and other allied phenomena in biological organisms called systemic based approach.

1.2.2 Project Initiatives

Literature study showed that there is no system as yet which looks into the designing and assessing the protection scheme for a power system. Many design companies have a wealth of knowledge and expertise which they utilise daily for the purpose of decision making [Lai 89]. However, this knowledge is rarely recorded explicitly and even if there is recorded data, access to it is often cumbersome. This introduces complications for a company when the expert staff retire or move to other employment. The knowledge and expertise of the expert staff are no longer readily available as expected. Further problems arise where non expert staff will have difficulties in re-learning something from first principles or locating information which could be dispersed everywhere. Thus, time, effort and consequently money is wasted in gathering information, knowledge and expertise whenever it is required [Lai 89]. Therefore, to avoid this wastage and related problems from cropping up, an intelligent system could be developed to record the knowledge and expertise of the experts and make it readily available any time, whenever experts advice or heuristics decision is required.

Furthermore, it has been noted that there exists a serious problem of shortages of expertise in this area of power system protection and/or are expected to stay in the organisation. This problem is expected to worsen in time to come. As one of the possible solution, a need for some sort of an intelligent knowledge based system is called for. This system to be built is expected to assist in management of changes in an organisation by preserving the expertise and minimise the effects of the loss of protection engineers, and also to retrain new engineers.

It is largely due to the above reasons that this research project is initiated. The aim of the research is to build an intelligent system which automates the process of designing a protection scheme for a power system. Feasibility study has to be made to select the most applicable methodologies and techniques to be used and to adopt an appropriate approach in designing the system architecture. In addition, a prototype is later built to ensure the successful implementation of the system.

1.2.3 Research Objectives

The system discussed in this thesis is an adaptive multiagent system. It has a distributed knowledge base and the domain knowledge is separated into several smaller units. These units of knowledge reside with a number of loosely coupled knowledge sources or software components, better known as agents. The division of domain knowledge into specialised areas gives the agents the necessary expertise; the employment of multireasoning paradigms and strategies gives the agent the required skills - both the features characterise an agent as a specialised problem solver. In addition, the sophisticated inference mechanism is sufficiently advanced to reflect and imitate the expert's way of thinking. They also have the ability to select the most appropriate strategy for a task while switching between them whenever necessary.

The main focus of this research is on the study of agent architecture with a presentation of an approach to the design of the architecture which consequently brings to the development and implementation of the system. The domain area in which the system is applied to is in power system protection.

A power system consists of many components that need to be protected against occurrence of any series or shunt faults. Protection of a power system is mainly concerned with protection for all the components of the power system. The protection schemes available vary depending on a lot of factors viz. location and importance of the power system, the components to be protected, the economic constraints, the availability of resources, the authority's policies, etc. One of the biggest influencing factors is the availability of funds for the implementation of a protection scheme.

There have been a large and ever increasing number of research and development of expert system applications in power system [ESAP 93, EMPD 95, IEA/AIE 95, ISAP 96]. The areas of interest cover power system management, system restoration, forecasting, protection, communication, distribution, transmission, scheduling, monitoring, security, control and many more. However, the number of research projects in intelligent power system especially in the protection area is relatively small. These include application and simulation systems developed for setting calculations for distance protection in zones 2 and 3, load forecasting, distance relaying, alarm processing, fault analysis and transformer protection [APSCOM 91, Gora and Kacejko 89, Lee *et al.* 89a, Lee *et al.* 89b, Kalam and Negnevitsky 93, Liu *et al.* 94].

The generic agent architecture design can be applied to build a wide range of intelligent or decision support system applications. Problem solving by agents is handled sufficiently by employment of multireasoning paradigm. The different agents in the system include a coordinating agent, design agent and an interface agent. The coordinating agent is responsible for decomposing a

problem into a set of subtasks which are distributed to the other problem solving agents (i.e., design agents). Instead of a bidding protocol [Genesereth and Ketchpel 94], decomposed tasks are assigned to the various agents based on their knowledge and expertise in the specific problem domain. Coordination and convergence of the sub-solutions into a single solution are achieved by the coordinating agent. The interface agent is responsible for communicating the system's response with the user. In other words, all communications between other agents with the user is accomplished via the interface agent.

This research presents a novel approach to the design of an agent architecture using the layered methodology based on the study of system requirements. The design is an incremental and adaptive process where the components of the system are developed and their existence is justified with the environment and task requirements before the components are integrated and put together to construct the complete architecture. Subsequently, the agent architecture is used to build different types of agents which are then applied to build intelligent multiagent systems. Three multiagent systems based on distributed knowledge base systems are designed. They are then prototyped, studied and compared to enhance the understanding of the compositions of the underlying structure. The implementations are carried out in different environments such as knowledge base and object oriented environments.

The application area chosen for the implementation of the multiagent systems in this thesis is power system protection. Power system protection is selected because it represents a sophisticated class of engineering designs. The system is intended to assist protection engineers in the design, selection and analysis of protection schemes for a power system. The design of a power

system protection is not a simple task; it is tedious work which requires a great deal of expertise and experience. The knowledge of a professional practising protection engineer cannot be acquired from just texts and reference books only and the decisions that have to be made are usually subjective and heuristic. The integration of different methodologies and techniques is used by the system to aid the protection engineers in decision-making especially in the design and assessment of a protection scheme. In addition, the system can be taken as a teaching tool that will provide some training exercises to the inexperienced engineers in the protection area.

1.2.4 Original Contributions

The construction of the intelligent agent is based on the generic and adaptive agent architecture. The result is the development of intelligent agents that are capable of solving problems which may require heuristics and subjective decision making. These agents are then organised in a variety of ways to build intelligent multiagent systems.

The architecture of the multiagent systems which employs different types of agents have been designed using a new approach to the study and design of architecture. The multiagent together with the multireasoning paradigm is used to address the design and coordination of protection schemes for a power system. The system is definitely new in the application domain and has not been applied in the area of power system protection.

This thesis makes the following contributions :

a) It makes use of an innovative and adaptive approach to the modelling and design of a generic architecture for intelligent agent;
b) The application of agent technology and multiparadigm reasoning techniques in the development of an intelligent multiagent system is new in the area of power system protection;
c) The concept of automating the process of designing a protection scheme for a power system is original;
d) The separation of the case library/case memory into primary and secondary memories helps to generate reliable outputs and improve the efficiency of the system performance and search time;
e) The index of a case encodes the key features of the case which is used to interpret a case and used as a basis to retrieve cases of the similar set of features.

1.3 Thesis Overview

This thesis is divided into the following four parts:

- Part I consists of chapters one and two which presents the problem that needs to be solved.
- Part II consists of chapters three, four and five. This part make up the main theme of the thesis and the work that is carried out to solve the problem.

- Part III which contains chapters six, seven and eight presents the organisation of the intelligent agents in various ways to model and design different multiagent systems' architectures. The systems are applied to the same application domain but implemented in different system environments. Subsequently, evaluations are carried out and presented in chapter eight.
- Part IV concludes with a review of the thesis and closes with a scope for future direction in chapter nine.

Chapter one gives an overview of literature, an introduction to the research initiatives, outlining the research objectives and the contributions of the project work.

Chapter two reviews the agent technology, distributed artificial intelligence and power system protection. Multiagent systems are discussed with a review on the development of some systems that have applied various approaches in diversified fields. A brief technical background on power system protection is given, which was felt to be necessary as the application domain of this research is in power system protection. This is further followed with a review on the systems developed in power system protection.

Chapter three gives a brief review on the study of architecture and the importance of architecture design. A short description on the design based approach introduced by Sloman [Sloman *et al.* 94] is also presented.

Chapter four presents an innovative approach to the design of an agent or system architecture. An architecture is the base structure and the essence to a

robust and efficient system. Therefore, the need to emphasise and conduct the study of architecture arises. Even though there have been some research in system architectures carried out by The Knowledge Systems Lab at Stanford University [Hayes-Roth 94, Hayes-Roth *et al.* 94] and The Cognition and Affect Group at University of Birmingham [Read 93, Read 94, Read and Sloman 93, Sloman 92, Sloman *et. al* 94, Sloman 95], the overall growth of interest in the area is slow. This chapter introduces a different approach to the design of an agent/system architecture and follows on to the study of architectures with the discussion and comparison of the alternatives to the architecture.

Chapter five expounds on the application of the architecture presented in the previous chapter to the development of an intelligent agent. This chapter details the design of the agent architecture with the aim of developing a generic, adaptive and intelligent agent for the construction of multiagent systems which are applicable to a wide range of problem domains.

Chapter six presents three possible multiagent systems' architectures which were designed by reorganising and redefining the responsibilities of the agents in the systems. The three systems are a distributed knowledge base system with centralised control, a distributed/shared knowledge base system with centralised control and a distributed knowledge base system within a federated system framework.

Chapter seven presents the application of the systems to power system protection and the eventual implementations of a prototype for each of the system architecture which has been designed in chapter six. The environments under which the systems were implemented include Lucid Common Lisp 4.1 (a

knowledge base environment) and O_2 (an object oriented environment) running on SPARC Sun OS 4.1.x under Open Windows 3.0. In the first system environment, two additional libraries of Common Lisp subroutines were loaded into the working environment - Epilog and Api. (Refer to *Appendix A* and *B* for further information on Epilog and Api packages).

Chapter eight presents the evaluations on the systems' implementations, studying the advantages and limitations of each system architecture.

Chapter nine gives a discussion on the evaluations carried out and reviews the completed work on chapter basis. It concludes the thesis with suggestions for future research.

Chapter Two

LITERATURE REVIEW

Abstract

There have been more and more intelligent systems built nowadays, using agent technology and distributed problem solving technique. These technologies have often been used and applied in the development of the more complex and complicated systems. Many have concentrated in building a generic architecture that could be reused to build similar systems in various domains. However, as more intelligent systems are applied to many diversified areas, there are still many interesting areas that are yet to be explored. One of these areas is power system protection. This research brings together the popular technologies with multiparadigm approach and their application to power system protection. Building an application system is not as easy as it seems - there are many issues to be resolved. Such issues begin with getting the system architecture correct - questions that need to be answered include determining the system's requirements, main components of the systems, and the type of paradigms or techniques that would be employed. These issues will be discussed in detail in the subsequent chapters. This chapter reviews on agent technology, multiagent systems and power system protection; and the current application systems being developed in the protection area.

2.0 Introduction

There are increasingly more developed systems and research projects in intelligent systems using distributed problem solving technique and agent based architecture [AAAI 94]. These methodologies which include distributed

problem solving together with Artificial Intelligence (AI) techniques, and agent technology seem to be the present and future directions for developing intelligent systems in many diversified areas including medical, manufacturing, diagnosis, networking and engineering [Wielinga *et al.* 92, Huang and Brandon 93, Jennings *et al.* 93, Khedro *et al.* 93, Huang *et al.* 94, Peligry 94].

Recent attempts to develop larger and more complex knowledge based systems have revealed the shortcomings and limitations of centralised, single expert system architectures [Adler *et al.* 92]. The latter type of architectures are known to be monolithic, inflexible and can not adapt to changes. There are very few distributed applications and intelligent systems in the power area [Huber and Wu 91, Ngan *et al.*. 91, Sriyananda and Silva 91] and the growth of such applications in the area is rather slow [APSCOM 91, AUPEC 94, MS 93, ISAP 96]. Furthermore, these systems have been only to serve as an integration unit for a number of geographically distributed systems or different components within a system. The functions of these systems basically involve data exchange and monitoring - the reasoning technique employed is mainly inductive reasoning.

As a response to the current problems to overcome the shortcomings of the more traditional systems which are too monolithic and rigid, we have come up with an adaptive intelligent system using agent technology and distributed problem solving technique with multireasoning paradigms incorporated in the knowledge base. The system developed in this thesis is an adaptive system which makes it superior compared to the traditional systems:

- In the operating sense, it can adapt itself to the dynamic environment and provide appropriate solutions to the ever changing requirements of the user;
- In the implementation sense, it can be easily adapted and expanded to accommodate other modules and/or agents.

2.1 Agent Technology and Distributed Artificial Intelligence

The term 'agent' has become a popular keyword and concept in the field of research nowadays [Adler *et al.* 92, Huang and Brandon 93]. It is often applied in the development of the more complex and complicated systems. Such systems are usually decomposed into smaller subsystems which are then made to interact or communicate with each other for the purpose of cooperating to solve problems. These subsystems which may be distributed logically or geographically, could be built and represented by agents. These agents are often referred to as distributed artificial intelligence (DAI).

Distributed artificial intelligence is a sub-area of Artificial Intelligence which is also referred to as cooperative Distributed Problem Solving (DPS) [Uma *et al.* 93]. DPS is concerned with the application of AI techniques and multiple problem solvers (ie. agents) [Decker 87]. DPS is characterised by the existence of interdependencies between the decomposed tasks leading to a need for the agents to cooperate extensively during problem solving. It involves distributing control and data to achieve cooperation, coordination and collaboration among the agents which are necessary to solve a given problem. Distribution implies the decomposition of a problem into a set of sub-problems

or tasks to be solved by multiple processing units such as CPUs or agents [Oates *et al.* 94]. There are many forms of cooperation, such as *task-sharing* and *result-sharing* [Smith and Davis 81]. Cooperation as discussed by Smith and Davis [Smith and Davis 81] is a method used when each agent has different knowledge and is an expert in its particular domain. Most of the agents in a distributed system has sufficient knowledge to generate at least a partial or a sub-solution. Therefore, cooperation or coordination among the agents is inevitable and necessary to solve a problem. Cooperative frameworks can also minimise communication, allow load balance, distribute control and knowledge, while maintaining coherent behaviour [Decker 87, Smith and Davis 81].

Agents have only partial and incomplete global views of solution requirements and the state of problem solving. Consequently, they arrive at partial solutions which are then interacted and coordinated to form the global solution. Tenney and Sandell called this independent decision agent with a model only of some subsystem on which it is an "expert" as *domule* [Tenney and Sandell 81]. One of the most important feature about agent that helps to achieve results in the presence of complexity and uncertainty is cooperation. Cooperation between problem solvers or agents must be structured as a series of carefully planned exchanges of information (i.e., communication) [Uma *et al.* 93]. Communication is important as an individual agent has no knowledge of the structure of the system outside its domain and it is only through communication that some desired overall performance could be achieved. Performance, on the other hand, depends on the problem solving architecture [Uma *et al.* 93]. Therefore, it is appropriate to consider carefully the frameworks or architectures and the necessary strategies required for cooperation when developing any system. Examples of some of these frameworks are contract net,

blackboard model, distributed, parallel blackboard models, scientific community metaphor and organisational structuring [Uma *et al.* 93].

An agent based system can be represented by a collection of agents whose objectives are to solve any problems for which they have sufficient knowledge [Gaiti and Pujolle 92]. There are basically three main types of agents architecture that have been identified and developed over the past years :

i. Autonomous agents;
ii. Agents which surrender their autonomy to another agent;
iii. Semi-autonomous agents.

The autonomous agents are the independent entities which usually maintain their own knowledge base. They have total and equal control, and participate in group problem solving and negotiations to reach an agreement [Bussmann and Muller 93, Genesereth and Ketchpel 94]. They are the competitive agents which bid for a job or a task. Direct communication is achieved via message passing or broadcasting through the use of a blackboard. These agents spend majority of their time communicating with one another for knowledge or information exchange. More importantly, they have beliefs, revisions and they can perceive, reason and act in a cooperative manner to achieve multiple goals in dynamic, uncertain and complex environments [Hayes-Roth *et al.* 94, Sycara 94].

In a federated system, the agents relinquish their autonomy to a representative or a higher level agent (i.e., a facilitator) [Devapriya *et al.* 92, Khedro *et al.* 93]. The agents do not communicate directly with one another but

rather through their local facilitators which take up the responsibility to fulfil their needs. The agents send application-level information and request the facilitators which transform them into application-level messages. These messages are then routed to their appropriate destinations. Cooperation and coordination are handled through the communication and cooperation between the facilitators. The common mode of communication in this type of architecture is through message passing, even though blackboard is also used at times.

In system architectures where semi-autonomous agents are employed, an agent usually involve other agents in its own problem solving tasks, for example, request for information or for tasks to be performed [Maes 94, Wittig *et al.* 94]. Collaboration and communication between various agents, according to Lashkari *et al.* [Lashkari *et al.* 94], takes two general forms:

a) desperation based communication where an agent seeks the help of other agents because of its own insufficient experience to make a confident prediction;
b) exploratory communication which is initiated by agents in bid to find the best set of peer agents for help in certain classes of situations.

According to Durfee *et al.* [Durfee *et al.* 85], there are three important approaches to improve coordination among cooperating nodes (agents):

- multiagent planning;
- negotiation;

- functionally-accurate, cooperate (FA/C) approach.

In the multiagent planning approach, the planning node forms a multi-agent plan that specifies the actions that should be taken and distributes them among the nodes. The disadvantage of this approach is that achieving the global view of the problem for the multiagent plan could be time consuming and communication intensive.

The negotiation approach concerns the decomposition of a problem task into a set of sub-tasks by a node which are assigned to other nodes based on a bidding protocol. However, the disadvantage of this approach is that a most suitable node for a task may not be available to bid for the job because it is occupied with another previous assignment.

The FA/C approach is defined as nodes which cooperate by generating and exchanging tentative, partial solutions based on their limited local views of the network problem and eventually converge on an overall network solution. The limitation of this approach is that the nodes may require much more time to converge on a solution as they may be working at cross-purposes.

2.1.1 Multiagent System

Why adopt a multiagent system approach or apply agent technology as distributed artificial intelligence or distributed problem solving technique? From a cognitive science viewpoint, Decker [Decker 87] quoted Hayes:

"All real systems are distributed."

Truly intelligent systems may contain so much knowledge and information that they must be broken down into multiple cooperating systems to be feasible. The participating systems are hence the distributed problem solvers that support collaborative tasks between human and computer agents [Decker 87].

Distributed problem solvers are referred to as a set of cooperating agents dividing the work to solve a problem and multiagent systems are referred to as a number of agents coordinating to solve problems. While the former takes the divide-and-conquer approach, the latter is an emergent system where the whole is greater than the parts.

Distributed artificial intelligence can be applied to homogeneous or heterogeneous systems. Homogeneous systems are systems that use agents of a similar type or with similar knowledge within the same platform, whereas heterogeneous system would involve integrating and coordinating a number of pre-existing or new systems from different platforms. Heterogenous agents may differ in their knowledge content, representation and application of their knowledge. For example, agents could represent the same knowledge differently to optimise their particular use.

Agent technology applied as distributed problem solving technique offers advantages such as speed, reliability, extensibility, ability to handle applications with a natural spatial distribution, ability to tolerate uncertain data and knowledge, modularity, conceptual clarity and simplicity of design [Smith and Davis 81, Reddy and O'Hare 91]. In addition, the high modularity of such

systems ensure conceptual clarity and simplicity of design. A good example of a multiagent application is CIDIM (Cooperating Intelligent Systems for Distribution Management Systems), which is aimed to help control engineers to manage electricity distribution and supply networks [Cockburn and Jennings 94]. CIDIM is one of the recent system developed that has applied agent technology in the area of power management. Examples of some other application systems that have been implemented successfully in many areas include those in industrials, medical, speech recognition, design, planning, concurrent engineering and cooperative information gathering [Huang *et al.* 94, Hayes-Roth 94, Peligry 94, Oates *et al.* 94].

Large systems are normally built in a distributed fashion to overcome and master complexity of the application domain. With ever increasing amount of information available and required in problem solving in this modern world, such systems would be very useful in integrating the decision making of humans and machines. According to Wittig *et al.* [Wittig *et al.* 94], this should ideally separate the control and execution process which makes the control part more explicit. This is where systems of nowadays differ from the older generation systems. The latter are the single large monolithic systems which are commonly known to be complex, brittle and inflexible. These systems usually consist of one large centralised control where problems are dealt with centrally in a non-distributed, monolithic manner.

Cooperation in multiagent systems is necessary, if not inevitable. Cooperation could be defined as agents working together to improve their individual performance or the collective performance of the system. According to Adler *et al.* [Adler *et al.* 92], there are about six strategies for cooperation:

(i) the system designer hardwires the translation of information and knowledge exchanged among the agents and directs all necessary traffic;

(ii) a globally accessible data structure provides information to perform necessary translations and deal with the generated traffic;

(iii) a globally accessible data structure contains only the universally understandable information and knowledge;

(iv) agents themselves perform all necessary exchanges directly;

(v) agents negotiate and converge on decisions by making deals under various types of pressure;

(vi) agents converge on decisions by making deals using probabilistic methods.

The system implemented in this thesis consists of semi-autonomous agents and are organised in a hierarchical structure. The control and execution are distributed to a coordinating agent and problem solving agents respectively. Message passing paradigm is used as the form of communication. Coordination is resolved by the coordinating agent which is also responsible for decomposing a problem and assigning the decomposed tasks to the appropriate problem solving agents. One of the merits of a hierarchical structure over autonomous agents is that there would be less communication back and forth among the agents and hence, time could be saved.

2.1.2 Review of Systems

Huang and Brandon [Huang and Brandon 93] developed AGENTS system to demonstrate the use of essential constructs and strategies for communication. AGENTS is a domain-independent general purpose Object Oriented Prolog language for cooperating expert systems in concurrent engineering design. The approach is based on distributed and cooperating knowledge based expert systems. Its agent has the standard structure of knowledge based systems, that are, an inference engine, a knowledge base and a global working memory shared by all agents. The global working memory is implemented using a blackboard, which records decisions and their relationships.

ARCHON [Wittig *et al.* 94] is a multiagent architecture which facilitates cooperative problem solving in industrial applications. ARCHON emphasised on loose coupling of semi autonomous agents to increase cooperation among the agents or pre-existing computational systems. ARCHON is a layered architecture; each layer is a knowledge based system reasoning about its domain system and the coordination within the community, but not solving any of the domain problems.

EMMA (Enterprise Modeling and Management Architecture) [Sycara 94] is another distributed artificial intelligence system, which facilitates information dissemination and cooperation of heterogeneous functions of an enterprise. This is similar to ARCHON and it too has a layered architecture. The functionalities and protocols of the system are divided and provided by the six layers in EMMA. Each layer relies on the functionalities and protocols provided by the previous layer. EMMA [Sycara 94] agent consists of four subsystems: (i)

Problem Solving Subsystem; (ii) Knowledge Base; (iii) Knowledge Base Manager; (iv) Communication Manager. The Problem Solving Subsystem consists of the inference engine and decision models which allow the agent to solve problems related to its domain of expertise. The Knowledge Base uses semantic inheritance networks and frame representations to maintain descriptions of various entities - physical or conceptual objects. An agent may use other agent's knowledge, in which case, a copy of it may be stored locally. The Knowledge Base Management system is responsible for managing information exchanges between the Problem Solving Subsystem and the Knowledge Base, maintaining the consistency of the local knowledge base, and responds to other agents requests. The Communication Manager consisting of a searcher module and a responder module, manages search for information in the system and responds to other agents. In comparison, the generic agent architecture developed here is also multilayered. However, each of the layer is constructed with the intention to raise the agent's intelligence to the level mimicking that of a human expert. Furthermore, our agent's knowledge base is efficiently organised into domain knowledge and inference knowledge. The latter is designed into sublayers to contain multireasoning paradigms. The former is divided into two layers - facts base and case library[1].

The agent architecture described by Huang *et al.* [Huang *et al.* 94], comprises of multiple layers of knowledge, a working memory, a communications manager and a human-computer interface. The agent knowledge is divided into three layers, which are the domain, inference and control knowledge. The control knowledge applies inference knowledge to the domain knowledge in order to generate inferences whenever new data is added

[1] The agent architecture is detailed in chapter 5.

to the working memory. Communications among agents are achieved by message passing via the agent's communications manager.

Jennings *et al.* [Jennings *et al.* 93] used distributed artificial intelligence technique to incorporate two stand-alone and pre-existing expert systems into a community of cooperating agents for diagnosing faults occurring in real particle accelerator process. This is achieved using GRATE (Generic Rules and Agent model Testbed Environment) to control the cooperative activity. GRATE is a general framework for constructing communities of cooperating agents for industrial applications. The agents have two major components - a cooperation and control layer, and a domain level system. The communities control is completely distributed - there is no hierarchy and no global controller. All agents are equal and have a degree of autonomy in generating new activities and in deciding which tasks to perform next. The inference employed is a forward-chaining process while the communication is achieved using message passing paradigm. It utilises the FA/C paradigm where the agents asynchronously exchange partial results about their processing stage to ensure consistent interpretation of the whole problem by the entire community.

Nishiyama *et al.* [Nishiyama *et al.* 92] incorporated Petri Net into a multiagent system to provide a framework for multiagent planning. The agents' plans are integrated into a consistent total plan using constraint satisfaction approach and are represented as an occurrence net, also known as a Predicate/Transition (P/T) net, a kind of high level Petri Net. In this framework, objects are introduced as passive components without actions, while agents are defined as active components executing their actions. Actions are regarded as state transitions with goals or intentions. Based on these notion, Nishiyama *et*

al. regarded a plan as a sequence of actions by agents, an envisionment as a sequence of state transitions by objects and a constraint as a sequence of state transitions by system components. A few scenarios were presented to show how the agents deal with envisionments and plans as constraints. The communication process between the agents and objects is implemented using Occam, a parallel processing language.

Petri Net is also applied to develop a framework for the resolution of FMS (flexible, programmable, integrated automation) control problems [Devapriya *et al.* 92]. Devapriya *et al.* used a Petri Net based kernal system to support the reasoning within the blackboard system architecture. Objects which could represent machines, transporter, etc. are organised into problem solving nodes. Each node consists of a set of problem solving agents where communication is achieved through broadcasting and message passing. Jobs are assigned to agents using the bidding protocol (i.e., the lowest priced bid by an agent wins the job).

Khedro *et al.* [Khedro *et al.* 93] introduced an agent based framework for the development of integrated facility engineering environments in support of collaborative designs. Design agents represent various existing design and planning systems that communicate their design information and knowledge partially and incrementally using the Agent Communication Language (ACL)[2]. The communication of the design agents is coordinated via system programs or known as facilitators in a federation architecture. One of the main advantage of this federation architecture is improved flexibility in integrating agents, which

[2]ACL is a formal language proposed as a communication standard for disparate software. It is based on a logic based language called Knowledge Interchange Format (KIF) and a message protocol called Knowledge Query Manipulation Language (KQML).

allows different agents to be ignorant about other existing agents in the architecture. Message passing paradigm is used as the form of communication and they are routed to the appropriate recipients by the local facilitators. The facilitators also perform other responsibilities which include facilitating communication and exchange of design information and knowledge among agents, translating messages, scheduling the executions of different agents, assisting in decomposing complex problems and relating different design information.

In DARES [MacIntosh *et al.* 91], a distributed automated reasoning system, the agents are built with incomplete knowledge about the state of the world. A cooperation strategy which is dependent on the initial knowledge distribution was developed to coordinate the semi-independent agents.

Shaw and Fox [Shaw and Fox 93] describe two implementation examples of decision support systems (DSS's) for aiding group problem-solving situations. The first system NEST - networked expert systems testbed, is a prototype system consisting of four expert systems. The second system describes a group decision support system (GDSS) for design fusion system. Both systems illustrate a multiagent problem solving system based on the blackboard architecture. The expert systems communicate via a blackboard or mailbox for coordination.

2.2 Power System Protection

2.2.1 A Brief Technical Background on Power System Protection

An electric power system represents a very large capital investment. To maximise the return, the system is loaded as much as possible and is usually in full operation continuously [GEC Alsthom 87, Ravindranath and Chander 77]. A power system should ensure the availability of electrical energy without interruption to any load connected to the system [Ravindranath and Chander 77]. A power system may consist of a few kilometres to several thousand kilometres of distribution or transmission lines. Components of a power system include buses, lines, motors, generators, transformers, etc. Many of these components are exposed to the environment and are subject to damage and accidents due to external factors. In particular, these external factors which are responsible for the damage and breakdown of components include storms, lightning, corrosion and falling structures. These incidents would not only cause electrical faults but also mechanical damages. The most serious consequence of an uncleared fault is fire which may not only destroy the equipment at its origin but may spread to other parts of the system, causing total failure and endangering lives. Short circuit may be responsible for a complete breakdown of electrical supply, damage to the elements of the system including failure of relays and electric motors.

The electrical power technology throughout most countries in the world has been progressing steadily for decades now. It has reached the maturity stage where the construction of power stations are designed so carefully in order to fulfil the requirements of power stations to generate, transmit and distribute

electricity to its customers with maximum reliability and minimum cost possible. But no matter how well a power system is designed or constructed, the risk of a fault occurring on any part of the power system can never be totally eliminated. This factor signifies the importance of power system protection.

As mentioned before, a power system is always exposed to the risk of a fault occurring which could be due to storms, lightnings, falling of external objects or damage to insulators. Such incidents would not only cause mechanical damages but also an electrical fault. A fault usually produces repercussions throughout the network. Therefore, the risk of a fault occurring, however small it may be, is multiplied by the number of items which are closely associated in the extensive system. As such, it is imperative to provide some sort of protection to the power system.

The development of modern technology applied to power system has been rapid, and therefore there is just as much that is needed for protection and control of the power system as well. It is not only important but necessary for the design and the application of protection and control, to keep pace with the construction of power system which is becoming more sophisticated nowadays.

Protection for power system is a very large and important area that requires expertise in the management of changes and complexity, more so, whenever a problem or a fault occurs. A fault or any abnormality can be intolerable and unaffordable as well as it can be extremely dangerous and risky, not to mention the fact that a high cost will be incurred for repairs too. The function of a protection system is to minimise the effects of faults, if and when it occurs. This means minimising the dangers to humans and environments, reducing losses

and damage to the components of a power system and minimising interruptions of power supply to consumers. There are many different type of protection schemes. Each scheme depends on factors such as geographical location of the power system, the importance of the power system, the type of components to be protected, economic constraints and utility policies. The design of a protection scheme requires a great deal of expertise and experience on the part of the engineer. Design decisions, are in the main, subjective and follow '*rules of thumb*'. The protection scheme must provide proper and adequate protection to all parts of the power system and must be designed so that the system meets the reliability requirements, speed and selectivity as set by the system operating constraints.

Power system protection can be divided/designed to cover the different parts of the power system such as protections for motors, generators, transformers, busbars and feeder lines. There are different types of protection that could be applied when considering protection for a component, that is, overcurrent protection, differential protection and distance protection. The main function of the protection system is to isolate faulty network parts as quickly as possible. For example, when a power line which is connected to earth is broken, an earthfault is said to have happened. In such situation, the main protection has to operate on both sides of the line and send trip signals to both circuit breakers[3] in order to isolate the fault. However, if the main protection system failed to clear a fault, the backup protection which operates with a time delay should then open the circuit breaker to clear the fault [Wagenbauer and Nejdl 93].

[3] Circuit breakers is one of the protective device which can be operated either manually or automatically. They are installed in power circuits to open under normal or fault conditions so to isolate the fault from the rest of the power system.

These protections work on different concepts depending on the operating principles of the relays. However, where applicable, they could be applied jointly or separately for reasons of reliability, selectivity, costs, speed and discrimination. In any case, a protection scheme is very important to provide the most effective protection for a power system for reasons stated above. This is where the role of the protection engineers comes in - selecting the most suitable relays with the most applicable features to be applied to the power system. In short, different types of relays must be chosen correctly to be placed at various locations of a power system with the correct settings tuned so that the power system could be well and adequately protected. When a fault occurs, all the relays would start to operate. But it is only the relays within the 'faulted' zone are expected to send a trip signal to the appropriate circuit breakers to open. This will then isolate that part of the system from the rest of the system. The other relays will just reset.

2.2.2 Intelligent Systems in Power System Protection

There have been a lot of research going on in the field of power system. Most of these research concerns with the development and application of expert systems or decision support systems in most areas of power system. Such areas include alarm processing and fault diagnosis, security, planning and design, substation monitoring, and so on [Liu *et al.* 90]. Power system protection is one area which lacks focus all this while, but it is receiving quite a lot of attention lately. Even then, most of the expert system applications in the protective relaying field have been related to protective relaying, relay setting coordination function, alarm processing and fault diagnosis, steady state and dynamic

security, planning and design, system restoration, environments for operation aids, remedial controls, substation monitoring and control, and maintenance scheduling, and selection and coordination of fuses in an industrial customer environment [Lai 89, Kezunovic *et al.* 91, Gora and Kacejko 89, Lee *et al.* 89a, Lee *et al.* 89b, Hatta *et al.* 88, Russell and Watson 87, Taylor *et al.* 90]. There have also been lots of other simulation programs developed to serve as a training or learning tool and also as a mean to study and understand the measurements, characteristics and the occurrence of faults on various parts of a power system.

While there is a growing research interest in the applications of expert systems in power system protection, there is still not enough emphasis on the development of intelligent or decision support systems in this area. Expert systems are merely automated production rules systems which derived to the conclusion or solution from a forward or backward chaining inference process on the set of rules in the knowledge base. On the other hand, intelligent or decision support systems play a more supporting role in assisting the decision maker to select the most appropriate decision or course of action based on the problem constraints. The system may also examine and evaluate the different options available, and offer any explanations, if so required by the user. Intelligent or decision support systems are most successful in application domains where decisions have a more subjective nature and are based on heuristics and experience rather than on well-defined algorithms or analogical reasoning.

It is important that research efforts should now be more concentrated in the protection area of power system. Most of the protection system implemented

have been around decades ago. Emerging new technology have come up with new products and improved a lot of the protective elements incorporating multiple purposes in a single element, eg. microprocessor. However, it is too costly and prohibitive to replace the existing protection system with a new system unless the authority is building a new power system. Hence, as an alternative, the existing protection system could be assessed now and again. When an element breaks down, malfunctions or becomes inoperative, the action to be taken would be either to repair the element or substitute it with a new one. When it comes to substitution option, the exact same element must be purchased to replace the old one. Otherwise, another new different element (different manufacturer or additional features, etc.) could only be applied provided that the replacement has minimal effects on the rest of the system. Even then, the whole protection system may need to be reassessed again to ensure that the whole power system are properly protected. The settings of all the relays will need to be checked so that accurate desirable requirements are met.

The above task of determining whether an 'out of order' protective element or whether certain section of the protection system should be replaced or updated with new scheme is no easy task for a protection engineer. It involves analysis and reassessment of the whole protection system, study of the new scheme and how best to incorporate it into the existing system, the availability and cost of the new scheme and a lot of other considerations to be taken before recommendations of any decisions and actual work could be carried out. It is a tedious job of heavy responsibility and could also be a repetitive work too. Furthermore, if it is done manually, it could take a very long time before any suggested solution(s) could be presented to the 'authority board'. Besides, mounted pressures and time constraint could affect effective assessment work.

Hence, the importance of an expert system that could accomplish almost the same task but in a vastly reduced time. Besides that, an expert system could also help to alleviate the protection engineer's workload and pressure. This system does not takes over or assumes the job of a protection engineer but it acts more like an assistance. The eventual aim of having such system is to propose a few recommended solutions when presented with a problem based on its reasoning and knowledge base. From here onwards, the protection engineer will then have to do some evaluation among the recommended solutions which may be dependent on some other criterias or management policies to decide on the best option to take.

Most of the work done in the development and application of expert systems in the area of power system protection have been mainly concerned with real-time control of the environment. The majority of them use the rule-based systems. While a lot of the recent works are turning to the more newer approach and methodology in developing an intelligent system or a decision-support system, there are still a lot more work and research going on in this new area of interests.

2.2.3 Review of Systems

As stated in the previous section, the number of research projects in the development of knowledge based systems, decision support system and especially intelligent systems in the area of power system protection is very small compared to the number of research in the other areas of power system. This is indicated by the small number of papers presented in recent conferences

[MS 93, EMPD 95, ISAP 96]. However, there seems to be a significant increase in the interest of using AI techniques such as fuzzy logic, genetic algorithm, case based reasoning and a tendency away from the traditional rule based systems.

Knowledge based systems are becoming more popular nowadays. Kalam and Negnevitsky [Kalam and Negnevitsky 93] proposed a logical knowledge base approach utilising expert system techniques as an operational aid for overload clearance, while Marin and Banerjee [Marin and Banerjee 94] developed a knowledge based diagnostic system for the maintenance of transformers and circuit breakers.

The goal of a knowledge-base system is to encode the complex and sometimes changing knowledge of the human expert into a system and then to utilise the knowledge as an expert would. The authors stated that strong domain specific knowledge base is required in order to achieve outstanding performance [Russell and Watson 87]. This proved to be more so especially in area where experts are scarce and the situation is forseen to become even worse as in the area of power system protection.

Lu *et al.* [Lu *et al.* 88] have developed a knowledge-based tutorial and consultation system which is implemented in a personal computer, designed to aid both control center operators and Energy Management System software integration team members. It was concluded that with appropriate design, a knowledge base system, such as the one presented in the paper could become an excellent tutor for transferring knowledge from an expert system to his/her successor.

An interesting joint research project by Puget Sound Power & Light Co., WA and Electric Energy Group of Washington University is outlined by Sumic *et al.* [Sumic *et al.* 90]. The developed product is called Automated Electric Plat Design (AEPD). AEPD is a computer tool aimed at automating underground residential distribution (URD) design in new developments (also called electrical plat design). The AEPD tool was implemented using various new technologies including GIS (Geographic Information System), an Expert System and a Relational Database Management System - all integrated using an Intelligent Decision Support System (IDSS) approach. AEPD tool was described as a design tool in a Facility Management System, which manages the complexity of design in underground residential distribution.

The introduction by Lai [Lai 89] maintains the importance of expert system applications and the author has developed a prototype expert system for power system protection coordination to demonstrate and support his point. The expert system records and models the knowledge of power engineering experts in the domain of protection coordination for industrial power system. The output of this program is a design scheme for protection coordination of a power plant that meets its regulatory requirements. It was concluded that the system has proved to be successful, but however, the knowledge base needs to be expanded to include a larger portion of the application domain where the reliability and integrity of the system needs to be tested further.

Examples of some of the many expert system applications developed are in the protective relaying field of relaying and relay setting coordination function [Gora and Kacejko 89, Lai 89, Lee *et al.* 89a, Lee *et al.* 89b, Kawahara *et al.* 93]. Other programs developed include the calculation of impedances and

determination of settings for zones 2 and 3, a high speed digital distance protection scheme and analysis of transmission line insulation for lightning performance [Leung 91, Li *et al.* 91, Thum and Liew 91].

Literature survey shows there have been very few applications that focus on aiding the protection engineers in the aspect of designing protection systems. Research in this area has somehow been overlooked even though the applications would be very useful and practical in aiding the engineers in the tedious process of design and assessment of protection systems. Efforts to develop a system in this particular area has been made by Kalam *et al.* and Liu *et al.* [Kalam *et al.* 91, Liu *et al.* 94, Wong and Kalam 94]. Both are relatively very small systems which cover only the protection for transformer component of a power system, they also used the conventional methods in their system (i.e., rules).

The former, [Kalam *et al.* 91] is a knowledge based system developed using Personal Consultant Plus, which plays an advisory role and provides necessary protection scheme for a given power transformer setup. The latter, [Liu *et al.* 94] is an expert system built on OPS83 which plays a similar role for power transformers as well. Another interesting commercial application system under way is named CAPE (Computer-Aided Protection Engineering) and uses relations[4] to represent the domain knowledge in power system [Enns *et al.* 92]. CAPE is another research project developed under the sponsorship of ten major U.S electric utilities. It is a productivity tool and a computer-aided system that provides comprehensive computing and record-keeping environment.

[4]Relations here refers to relations between tables in a relational database.

2.3 Conclusion

From the literature study carried out, we come to the conclusion that power system protection is an important and interesting area to venture into. While there is a lack of initiatives in the development of intelligent systems in the protection area of power system, there is no existing agent based system in the area at all. These reasons have made the project theme an even more suitable and worthwhile research to invest on.

PART II

THE WORK

The Architecture Design of a Generic Agent

Chapter Three

APPROACHES TO KNOWLEDGE ORGANISATION

Abstract

The design of an architecture is very important, as it forms the foundation or core of a system. However, such study of architecture is not too popular. There is more to be studied than just designing - the designs have to fit into the operating environment and tasks. Most system architectures are designed without much considerations given to the system requirements; the system's components nor alternatives with respect to the architecture are not explored and compared. This may eventually lead to the failure of the system to operate efficiently or consistently. In fact, there are a few ways in which the study of an architecture could be conducted: (i) study the system architecture in their own right or as things evolved from earlier designs or (ii) they can be studied in a top down, bottom up or middle out manner. This chapter is an introductory chapter which discusses the needs and importance of architecture study. As a review, before the new approach is presented in the following chapter, one approach by Sloman[1], called design-based approach is presented here.

3.0 Introduction

Most systems' architectures are tailor-made to the application domain and hence, they tend to be satisfactory to the systems that are built on it. Therefore, it is difficult, if not impossible to evaluate the design of an architecture by comparing it with another architecture. While nowadays, generic architectures

[1] Refer to [Sloman 95].

are becoming more popular, it is still hard to compare one architecture with another. The main problem would be - on what basis should two or more architectures be compared on.

Seldom people would complain or criticise about the design of a system architecture if the system operates well enough and behaves in an expected and 'acceptable' manner. This may be true even though at times the system may not be an optimum or efficient system.

However, whilst there is no one perfect way to measure or compare one architecture with another, a system designer could still justify an architecture in other ways. One way is to analyse and compare a design with its alternatives. As stated by Sloman [Sloman 95] :

> *" No one design whether natural or artificial, can be understood fully if we don't know what difference it would made had the design been different in various ways. "*

There is more to be studied in designs - the design has to fit into the environment and fulfil the system requirements. There are many to study architectures for complex system. However, in order to understand an architecture fully, the functional roles and causal links of major components have to be known as well as how they relate to specific requirements. This means knowing the differences occurring to the system in the absence or presence of particular components in the system and how their existence affect the architecture and consequently the operation of the entire system.

An architecture, especially a generic one, could probably be considered or justified by *exploring* the components that are contained in the architecture. *Exploring*, in this context, means examining each individual component and study or scrutinise how it fits into the architecture and meets with the system's requirements at the same time; what is its contribution or functionality; whether the overall architecture can function efficiently or exist at all without the particular component.

The design of an architecture is justified if it can withstand the scrutiny. Most important of all, assuming other things being equal, the system to be built would be efficient, robust, reliable and meet with all the requirements.

Sloman advocated that the design of an architecture can be justified by studying the alternatives and how the major components relate to specific requirements. Sloman also quoted that [Sloman *et al.* 94]:

> *" Design decisions not derived from requirements can be justified for research purposes in various ways eg. by consistency with previous decisions, by empirical evidence concerning how people do things, ... "*

3.1 Study of Architecture

A single architecture without any explorations will only gives a shallow explanation and this is analogous to constructing a building without a strong framework. It is important to provide some explanations by showing or

indicating how the architecture is better or worse than other designs and revealing its trade-offs. This gives a better understanding of the design and the evolutionary pressures. Sloman *et al.* [Sloman *et al.* 94] called this comparing and analysing the sets of requirements and designs as '*exploration of design space*'.

Studying architectures has both scientific objectives which is concerned with understanding existing agents, and also practical objectives [Sloman *et al.* 94]. However, the task lacks definition because there is no general theory relating to the different types of architecture, and there is a lack of clear and unambiguous specifications of what is to be explained or modelled.

The design of an architecture is very important and must not be taken lightly. It is the foundation and the core or crux of a system. An inefficient, unstable or unpredictable system can be traced down to the problem of an ill-designed architecture. Therefore, an architecture has to be well designed and well understood in order to reap the following benefits and advantages [Sloman *et al.* 94] :

- useful, efficient and reliable machines can be built;
- problems involving complex system or interactions between different components would be easier to understand and manage;
- architectures could lead to improvements in educational procedures, counselling, social engineering, management (good practical solutions are based on good explanatory theories);
- the underlying architecture could show what sort of control is involved and what causes the interference.

3.2 Design Based Approach

The design based approach, as defined by Sloman [Sloman 92], do not assume that there is only one possible design; or that there are any absolutely necessary conditions to be satisfied; or that the notion of what is to be designed is precisely specified in advance. Sloman admitted that this design based approach is closely related to the 'design stance' as described by Dennett - which requires the designers to specify their theories from the standpoint of how things work. For example, how perception works; how motives are generated; how decisions are taken; how learning occurs; and so on. Design stance does not require unique solutions to design problems. Instead alternative designs with interesting, varied properties are explored to discover the possibility of other systems and how they differ.

As Sloman said [Sloman 92]:

> *"In true philosophical spirit we can let our designs, and our theorising, range over the full space of possibilities instead of being constrained to consider only designs for systems that already exist: this exploration of possible alternatives is essential for clarifying our concepts and deepening our understanding of existing systems."*

Design based approach, according to Beaudoin *et al.* [Beaudoin *et al.* 93] is a movement from specification requirements to design development and implementations that satisfy the requirements. Generally, the strategy used is a mixture of top down, bottom up and middle out approaches with the aim of

providing richer theories of explanation. Hence, the approach is more concerned with developing a set of functional requirements for the architecture of intelligent agents or systems. However, the problem area that generalises all areas or branches of AI is too huge to be covered. Since the design-based approach focuses on the mechanism tessellate where all intimate details of the mechanism are ignored, the problem area is thus reduced to a 'coarse-grained', global level, as stated by Beaudoin *et al.* [Beaudoin *et al.* 93], who asserts Sloman's statement: "*architecture dominates mechanism*" [Sloman 92]. Furthermore, it is also argued that the search for powerful explanatory architectures is more important than the search for algorithms [Sloman 95].

Progressing from requirement specifications phase to the design phase is not a straight forward task - it is an indeterminate and often an arbitrary process. Beaudoin *et al.* and Sloman [Beaudoin *et al.* 93, Sloman 95] have attempted to formalise this progress which resulted in the enumeration of six types of decisions regarding design. These decisions to be made are influenced by design requirements, constraints, empirical data, hardware and software requirements, experimentation with various options and test runs. In addition to the enumerated decision types, the formalisation process has extended the design based approach to systemic design. One of the limitations of design based approach is that useful theoretical knowledge is not generated. As such, systemic design which is a constrained subset of the design based approach, attempts to overcome this limitation by including considerations such as evolution, resource limitations and holism. Further details on systemic design are in reference of Read and Read and Sloman [Read 94, Read and Sloman 93].

3.3 System Requirements

Design is an art - there is no one absolute design solution to one system and often enough, there is more than one design which is possible and applicable. However, there is no general optimality measure and no unique mappings between niche space[2] and design space[3] [Sloman 95]. Designs always involve trade-offs and compromises. Therefore, an architecture should be studied and alternative designs should be explored in order to pursue maximum benefits and minimum cost, in addition to gaining the advantages listed in *Section 3.1*.

In designing an architecture for a distributed artificial intelligence (DAI) system, there are many considerations that must be given to the design and the possible mechanisms that could be used for problem solving which would satisfy the system requirements.

3.3.1 Distributed Artificial Intelligence - Design Considerations

A general DAI system consists of a group of problem solving agents collaborating in finding the solutions to given problems [Shaw and Fox 93]. In this framework, a significant amount of weights must be given in selecting the appropriate collaborating system, identifying the goal, organising the agents, distributing control and the coordinating mechanism for coordinating the problem solving activities.

[2]Niche space refers to the relationships between designs and various collections of requirements and constraints.
[3]Design space is the exploration of space of possible designs applicable for behaving systems.

There are basically three types of problem solving systems that have been developed [Shaw and Fox 93] :

- *collaborative reasoning systems* - agents in the system are involved in solving the same problem collaboratively. The main focus of this system is not on problem decomposition, instead, it is on guiding and coordinating the interactions among the participating agents so that the problem can be solved jointly by the group simultaneously;

- *distributed problem solving* - this system's activities are centered on problem decomposition, assignment of sub-problems to the various problem solving agents and finally synthesising and coordinating the sub-solutions into an integrated solution. Cooperation among the agents could be achieved through task sharing or information sharing;

- *connectionist systems* - agents are treated as the basic computational elements. Individually, they are not intelligent but together, they can solve complicated problems quickly.

Among the design factors that must be considered while designing a system are:

a. Goal identification

The problem solving process could be carried out in a deliberation procedure by all the agents collectively which is common in a collaborative system. Alternately, the process could be structured into four

steps: problem decomposition, task assignment, local problem solving and problem synthesis.

b. Knowledge distribution

There are different degrees of redundancy in the agent's knowledge base. The two extremes are:

(i) the agents replicate exactly the same knowledge;

(ii) each agent maintains an exclusive set of knowledge about the domain.

c. Agents organisation

The organisational structure of the agents determine the amount of information processed and the coordination necessary for the agents to operate efficiently. The structure depends on the responsibilities of each agent and the amount of coordination required and the autonomy held by each agent. For example, in a hierarchical organisation, the overall control could be held by one coordinating/controlling agent as opposed to a heterarchical organisation where each agent would have the same amount of control.

d. Coordination mechanism

In a multiagent society, coordination is inevitable to resolve conflicts, allocate limited resources, converge all sub-solutions into an integrated and coordinated solution. Shaw and Fox [Shaw and Fox 93] have categorised coordination into seven mechanisms :

- coordination by revising actions;
- coordination by synchronisation;
- coordination by negotiation;

- coordination by structured group mediation;
- coordination by opportunistic goal satisfaction;
- coordination by exchanging preferences;
- coordination by constraint reasoning.

e. Learning schemes

Learning should be included as an integral part of problem solving for improving strategies, updating knowledge and skills of the agents. This could be accomplished by data exchange, knowledge transfer or heuristics migration. Examples of some of these learning techniques that may be used are explanation based learning, case based learning and inductive learning.

3.4 Conclusion

This chapter has given an insight to the study of architecture design. It highlighted the importance of the study of architecture design as a good architecture will result in the development of a good and efficient system which will meet the system requirements. It has been known as a matter of fact that it is different to measure or evaluate a system architecture. And to design a system architecture without further explorations does not add value or weights to the system designed. Therefore, to ensure a system architecture is efficiently and adequately designed, a number of factors must be studied (e.g., goal identification, coordination mechanism, knowledge distribution, agent organisation and the degree of autonomy it holds, and learning schemes). In

addition to these factors, the alternatives of the system architecture should be explored and compared as well.

Chapter Four

A NEW ARCHITECTURAL DESIGN PARADIGM

Abstract

There is no clear unique concept of an intelligent system[1] and no unique set of requirements for intelligence because of the difference in the system requirements, operating environment and many other factors. Therefore, a fact that a system operates well does not necessarily mean that the architecture has been well designed. However, to evaluate one architecture by itself is a difficult task. There have not been many approaches developed to the study of architecture even though it has been identified as an important research area because the architecture forms the core foundation of a system. The correct and timely output, robustness, reliability and many factors of an operating system depends very much on the architecture. This chapter introduces a new approach - an incremental and adaptive approach in order to study and understand an architecture to be designed. This approach involves the study of the system requirements, identifies the major components of the system and, assesses them as they are added to the system. Once the design has been completed, other alternatives to the architecture are explored and compared.

4.0 Introduction

As stated in the earlier chapter, a single architecture without any explorations is unsatisfactory, just as it is not good enough to have a good design and implementation only. It is important to understand how a set of requirements is satisfied and why. According to Sloman *et al.* [Sloman *et al.*

[1] The word 'system' here is also used to mean agent as an agent can be regarded as an expert system.

94], there are many considerations to the various constraints on a design. For example, evolutionary constraints, physical environment, social structures, degree of dependence of neonates, the rate of change in the environment, and the availability of processing mechanisms. Thus, there is no clear unique concept of an intelligent system, and no unique set of requirements for intelligence because different control systems have different clusters of capabilities and requirements.

Most of the agent or system architectures have similar requirements for the main components (e.g., knowledge base, inference engine and/or control). Some intelligent systems in the area of monitoring may have additional modules such as sensors, behaviour, plans and emotions [Hayes-Roth 94, Hayes-Roth *et al.* 94]. The components that are employed in a system would depend on the system's requirements and problem domain.

For example, to build a small expert system, a knowledge base with an inference engine and a simple interface module may suffice. However, for a large complex distributed system, the components may include a multirepresentation of a distributed knowledge base, a separate inference knowledge which employs multiparadigm reasoning techniques, a control, a sophisticated interface module and probably incorporate built-in goals, perceptions, emotions to allow the system to behave more intelligently like a human expert.

This chapter presents a new approach to the design of an agent architecture for an intelligent system. The approach which is an adaptive approach begins with the study of the system requirements and correspondingly, the architecture

is designed in stages. As components are added to the architecture in layers, they would have to be validated - in regards to their existence, functionalities and contributions. The result is the design of an intelligent agent architecture which is multilayered. Different types of agents with different responsibilities can be then developed and organised in a variety of ways to construct different system design for a particular application domain depending on the system requirements and constraints.

4.1 Requirements of an Intelligent System

If a system should provide some sound or proven advice or response, then the system architecture should focus on the reliability and quality of the output produced. This is more true especially in some aspects of the medical and engineering fields. Consider for example, a system providing medical diagnosis and advice, or a system providing some engineering designs for constructions of buildings or power system. If the decision or advice given by the system is not reliable or have not been verified, then the output would not be of much value at all. Consequently, such systems would probably be futile or useless when implemented.

There has always been a lot of concentrated attention given to the system's knowledge base (i.e., how to acquire the knowledge, how to represent it, what should be in it, how should it be organised and managed, and how it is linked to other parts of the system). In AI, the goal is to develop working computer systems that are truly capable of performing tasks which require high levels of intelligence efficiently and effectively. In some systems, these tasks may

actually match or exceed human abilities. Nowadays, people are becoming more interested in intelligent systems which are capable of learning and improving themselves. Learning usually involves the expansion of the system's knowledge base. Besides efficient algorithms, an intelligent system would require a rich knowledge base to work and begin with. Knowledge is one of the main contributing factors to intelligence as Patterson said [Patterson 1990] :

"Food for intelligence is knowledge."

Knowledge can be defined as the body of facts and principles accumulated by human-kind or the act, fact or state of knowing. It has a familiarity with language, concepts, procedures, rules, ideas, abstractions, places, customs, facts and associations, coupled with an ability to use these notions effectively in modelling different aspects of the world. Knowledge combines relationships, correlations, dependencies and the notion of gestalt with data and information. In intelligent systems, the knowledge base is the central component where knowledge about the domain, its reasoning power and learning abilities are deposited. The intelligence level of a system depends on various factors such as the level and the amount of knowledge it possesses, how closely it can imitate the behaviour of a human expert, and so on. The next section discusses how the knowledge base is designed and organised to fit into the architecture of an intelligent system.

4.1.1 The Approach to the Design of an Intelligent Agent

The following illustrates the design of an architecture for an intelligent and reliable system where the correctness of the output is vital. Example of such systems include those for medical diagnosis, concurrent engineering, power system protection and many real-time systems. Sometimes critical decisions are made and actions are taken based on the output or response of these systems. Therefore, in these situations, the consequences of an incorrect output can be very costly, intolerable or even life threatening. As such, it is imperative that the output of these systems are accurate and reliable.

The following sections present the development of the major components for an intelligent agent, in order to build an intelligent and reliable multiagent system. As each components are identified and added to the agent architecture, they are evaluated and assessed (i.e., their fulfilment and contributions to the system requirements, their structure and their existence in the overall system).

4.1.1.1 Domain Knowledge

The Domain Knowledge can be viewed as divided into different parts [Wielinga *et al.* 92], each representing a certain viewpoint of the domain called *domain model*. A *domain model* is defined as follows by Wielinga *et al.* :

> *"A domain model is a coherent collection of statements about a domain that represents a particular viewpoint on the domain knowledge such that it is suitable for the*

> *problem-solving task. The domain model may therefore embody certain assumptions that are specific for the use that is made of it."*

As the complexity of knowledge based systems approaches the complexity of the real world, modularisation becomes imperative. Modularisation can be achieved by employing distributed problem solving technique using agent based architecture. Each agent's knowledge and expertise form part of the whole system's domain knowledge - thereby, making each agent a specialised problem solver. In other words, an agent is an expert system in solving problems related to its domain.

4.1.1.1.1 Case Library

In order to build a useful system which would produce consistent and reliable output, the knowledge base of the system should contain some actual past experiences or cases in addition to knowledge and references acquired from experts or other various sources. Therefore, when a case has been successfully retrieved, it means that the solution provided by the system would have been proven and verified already.

However, it is insufficient and do not make much sense to keep a library of cases which could only be retrieved if the case specifications (identified by salient features) match exactly the features of a specified problem. In most circumstances, there would be many cases found which are similar, but they do not match precisely to the current problem at hand. Hence, if only complete

matched cases are to be retrieved, then the probability of any cases retrieved at any one time would be very small, if at all. Consequently, to keep a case library which occupies a large memory space would be too expensive and inefficient to justify its existence.

Therefore, to improve the situation, the case library is separated into the primary and secondary sections. The secondary section is created to store cases which have been revised or adapted. That is, cases having close resemblance[2] to the problem are retrieved from the primary section. Once retrieved, these similar cases are modified and adapted to fit into the current situation to solve the problem. These modified cases are later stored in the secondary section to distinguish them from the proven cases kept in the primary section. The reason for the necessary existence of an additional secondary section is to allow the separation of the proven cases from that of the modified cases. This is because the latter type of cases have not been verified yet and have no history of success. Thus, they cannot and should not be treated as reliable cases.

Retrieval process would always be carried out from the primary section before proceeding to the secondary section, if necessary. The modified cases from the latter memory when retrieved, could be presented as solutions. However, they and any other cases that have been adapted should carry a '*warning*' that the solution has not been tested or tried before. It is therefore up to the human expert to test or evaluate the solution before applying them in real life situation. As soon as any of the modified cases have been verified and tested to be successful, the cases could then be promoted and transferred

[2]The definition of close resemblance depends on the algorithms used and on the application system.

automatically to the primary section. As an alternative, the secondary section could be linked to some other external programs that could verify those cases that have been successfully implemented or tested by a human expert.

Figure 4.1 shows the aforementioned feature incorporated into the case library. The expansion of the case library provides the system the ability to learn (i.e., through the growth of the knowledge base).

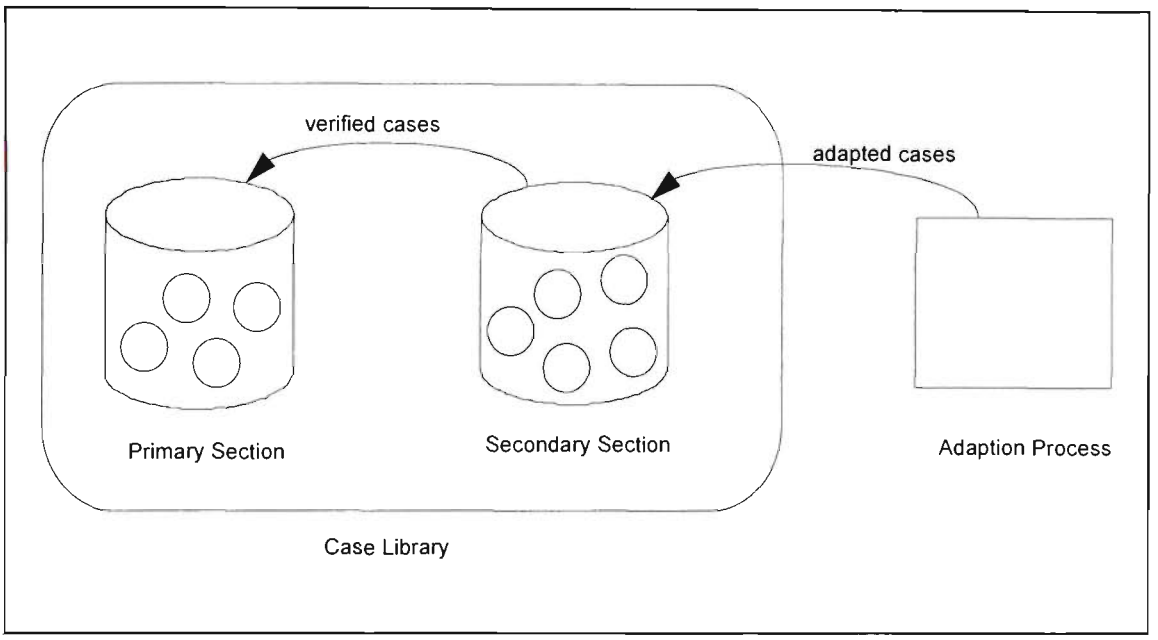


Figure 4.1 Expansion of Case Library

4.1.1.1.2 Facts Base

Having just the case library in the system's domain knowledge is definitely not adequate. There could be situations where there are no actual or similar cases found in the case library - either in the primary or secondary section. This may happen when a problem presented is new to the system and it has no experience in solving the problem before. Therefore to prevent the system from

failing or crashing, other forms of knowledge representations such as rules, facts and associations should be kept in the domain knowledge. This collection of knowledge is known as facts base. With different knowledge representations, various inference mechanisms or methodologies which include reasoning techniques such as rules, explanation based, goal based, fuzzy logic, causal reasoning, argumentation and functional based may then be used to produce a solution from first principle basis when case retrieval fails. Management on this part of the domain knowledge depends on the type of inference mechanism employed as different inference may require different form of knowledge representation.

4.1.1.2 Inference Knowledge and Domain Knowledge

The Domain Knowledge represents the agent's knowledge and expertise in a particular area. It contains facts and rules about the domain but it provides no further information about how the knowledge should be utilised. Therefore, a system with a rich domain knowledge but without the Inference Knowledge is like having a powerful sports car without any petrol - neither the system nor the car would operate or run. Similarly, the Inference Knowledge with built-in reasoners is required to specify the inference relations [Huang *et al.* 94]. For instance, the Inference Knowledge usually includes some algorithms and reasoning techniques which are actually the knowledge required, to be applied to task management and decision making.

The different inference mechanisms in the knowledge can be organised in layers[3] where interactions or communications between layers would have to be established. In a conventional layered system, procedure or function calls are made from one level to the next lower level. The more recent layered systems use message passing paradigm as the form of communication between the layers [Hayes-Roth *et al.* 94, Cockburn and Jennings 94]. The flow of communications back and forth could bring congestions and wasted resources if the traffic is not well managed. Therefore, instead of allowing the layers to communicate with each other, another possible arrangement is to have a selector in the Inference Knowledge.

The selector controls the inference mechanisms which act on the knowledge that may be represented in different forms in the Domain Knowledge. The main role of the selector is to select and invoke the appropriate inference mechanism during problem solving to manage a task. The selector as shown in Figure 4.2, sits on top of the layer which contains various inference mechanisms possessed by the agent. Depending on the current point during a problem solving process, the selector will select the most appropriate mechanism to deal with the problem and produce the best solution. In other words, the selector is responsible for the automatic switching between the various inference mechanisms to best solve the problem. This sort of arrangement and management is similar to the human behaviour which often switch between the logical reasonings that he/she possessed when solving a problem at hand.

The Domain Knowledge and Inference Knowledge are closely linked. The reasoning techniques in the inference knowledge influences and determines the

[3]Layered architecture will be discussed further in depth in the following chapter.

way the Domain Knowledge is represented and managed, and vice-versa[4]. For example, the Domain Knowledge is represented in the form of cases and consists of a facts base. Therefore, in order to be able to extract the applicable knowledge, the Inference Knowledge should contain some algorithms or techniques to search and retrieve the appropriate knowledge (or cases) from the case libraries or the facts base. If knowledge is retrieved from the case library, the cases may then need to be modified and adapted to the current situation. An example of one such technique that could be applied to this form of knowledge representation is case based reasoning. Alternately, causal reasoning, rules or qualitative reasoning can be used to apply to the facts base to produce a solution from first principles.

The functional separation of the knowledge base into Domain Knowledge and Inference Knowledge could mean that the Domain Knowledge is transparent to the reader of the knowledge base [Burstein *et al.* 94]. Figure 4.2 shows the interaction between Domain Knowledge and Inference Knowledge.

[4] Other determinants include implementation-dependent factors eg. the environment in which the system is implemented - in an object oriented or a knowledge base system.

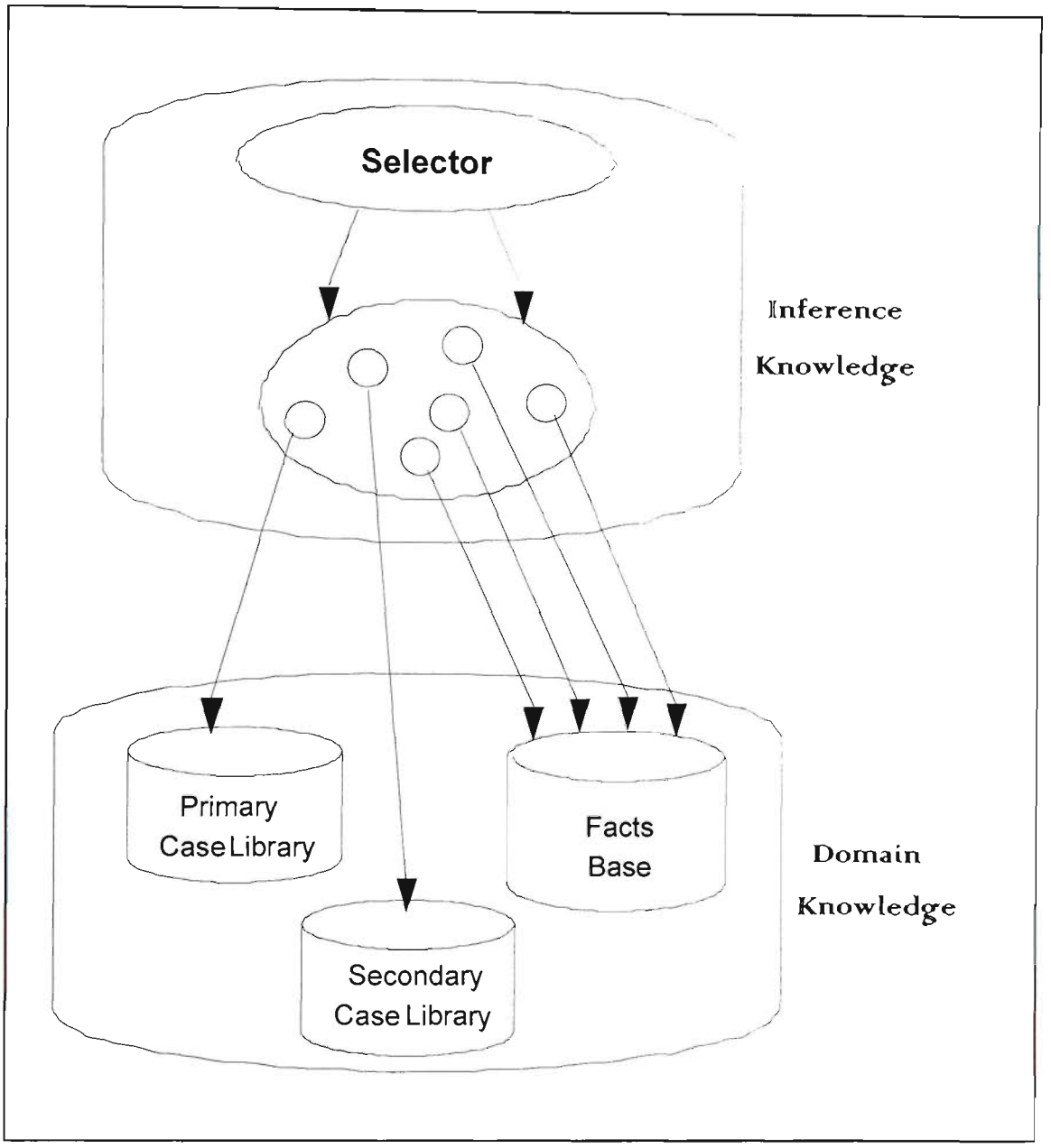


Figure 4.2 Domain Knowledge and Inference Knowledge

The separation of the Domain Knowledge from the Inference Knowledge promotes a number of advantages. Among them are: simplicity in representation and maintenance of knowledge; reusability is encouraged; acquisition and modifications to any part of the knowledge can be achieved independently without any side effects on the other part of knowledge.

4.1.1.3 Control

All systems require some sort of control. The control could be centralised or distributed - depending on the system environment and application domain. The Control is analogous to a driver - without the Control is like having the powerful sports car with a tank full of petrol but without a driver! The Control is required to direct the operations of the system effectively and efficiently. In term of the system organisation, the Control would be residing on top of the knowledge base.

The Control is expected to perform some management and controlling tasks which includes initiating a problem solving task, decomposing a problem, job scheduling, coordinating sub-solutions into an integrated and coherent solution. The Control interacts with other components within the system using message passing paradigm.

4.1.1.4 Interface Manager

While the Control has its specific and distinct duties to perform, it should not be burdened with other additional responsibilities such as managing the communication or interactions with other systems. In most developed systems, an agent or a system seldom, if ever, works alone. The agent or the system usually needs to interact with other agents/systems in either the same or different environment/platform, or otherwise, it may need to interact with the user. Interactions may involve more than just a simple exchange of information - it could involve communicating with another agent which may be internal or

external to the system (depending on whether it is in a homogeneous or heterogeneous platform). It could also involve interpreting and translating from one language to another and passing messages to the appropriate systems. Hence, for a system to function efficiently, an Interface Manager should be added to the top layer of the agent architecture.

The Interface Manager could be implemented in various ways depending on the environment it is developed in. It could be implemented as an optional layer residing on top of each agent. Alternately, the Interface Manager could be implemented as a separate but sophisticated interface agent that handles the interactions - all the internal and external interactions or communications of the system. In the latter implementation where an exclusive interface agent is constructed, the interface layer in each agent may not be necessary. This may be more appropriate when there is only minimum interactions existing in the system. Otherwise, it would be much neater for each agent to manage its own interactions via an Interface Manager, in addition to having a separate Interface agent as well. Such architecture may become necessary when implemented in a heterogeneous platform.

The Interface Manager in this agent architecture takes control of all interactions between the system and the outside world. It is responsible for communicating the internal requests/replies to the external source and interpreting/translating the external requests/replies to the system. As stated earlier, with the communication services provided by the Interface Manager, the other components in the agent could focus on their respective responsibilities and carry out their duties more efficiently.

Figure 4.3 shows the architecture which illustrates how each of the main components: Domain Knowledge, Inference Knowledge, Control and Interface Manager interacts with one another.

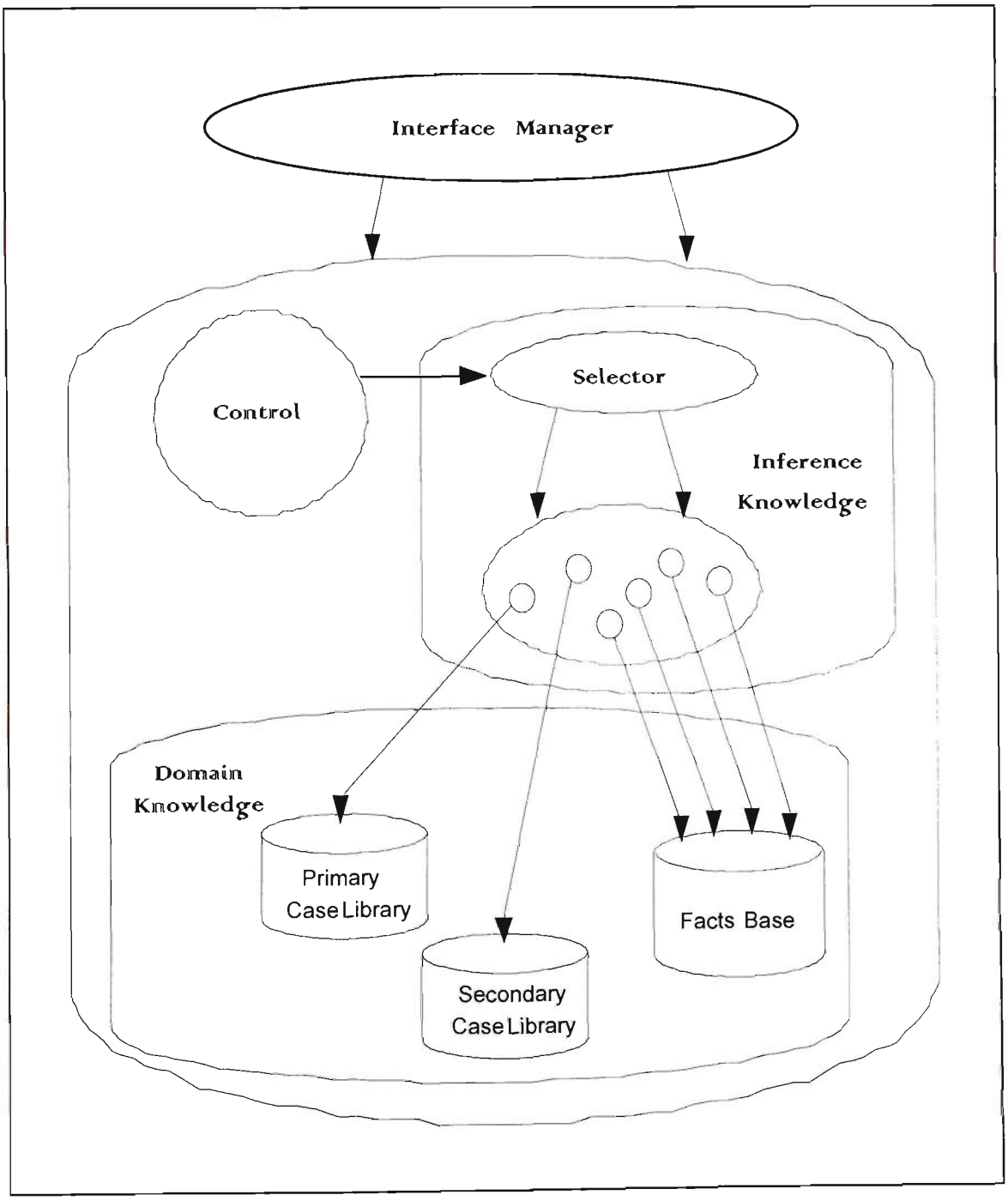


Figure 4.3 Generic Agent Architecture

4.2 Developing an Intelligent System

The following illustrates the application of the generic agent architecture shown in Figure 4.3 to the development of different types of intelligent agents to consequently apply them to construct a multiagent system.

4.2.1 Agents Requirements

The recent increase in the number of research of agent systems seem to strongly indicate that such systems are the answers to the limitations of traditional systems. Agent systems can also be best applied to domain problems which require a system to behave intelligently as outlined in the previous section.

Each agent is known as an expert system[5] - specialised in solving problems in its domain. According to Goodwin [Goodwin 93], an agent is an entity created to perform some task or set of tasks. The properties concern the suitability of various components of the agent can be defined in terms of success. The agent architecture follows the above approach, which in fact, takes the lead of the behavioural psychologists where the observable behaviour is the only criteria for determining success [Goodwin 93]. Such an approach enables us to avoid having to make inferences about the agent's belief, intentions or motives[6].

[5]The 'expert system' term here do not refer to the rule based expert system or production rule system.
[6]Therefore, agent's belief, intentions, motives will not be discussed.

The minimum requirements expected of an intelligent system are to be consistent, reliable and to be able to behave and response intelligently. At best, it should be able to simulate a human expert's decision making process including decisions that are subjective and heuristics in nature. Furthermore, it should be able to handle varied problems and provide the best solution, taking into considerations the dynamic environment and requirements of the application domain. In other words, the system must be able to update its own knowledge base to reflect the changes in the application domain.

An agent system usually employs several agents for different tasks. The common separation of tasks in a system include:

- interfacing and communicating with external sources;
- solving domain problems;
- coordinating the activities in the system.

Interactions between the agents are accomplished either by passing messages or through a blackboard system. If messages need to be interpreted or translated, it is performed by the Interface Manager of each agent. However, the overall communication between the system and external entities are carried out by an exclusive interface agent. The interface agent only performs interactions and communications activities. Solving domain problems are carried out by another type of agent, known as problem solvers while a coordinating agent would be responsible for coordinating the activities in the system.

4.2.2 Developing an Intelligent Agent

Two of the most important characteristics of an intelligent system are reliability and accuracy. The intelligent system is expected to deliver reliable solution or decision. In such circumstances, it would be suitable to have cases of protection schemes which have actually been implemented and proven, to be kept as part of the domain knowledge. As such, given a similar real-life situation, these schemes could then be retrieved and 'safely' applied.

There is much more to an intelligent system than just searching and retrieving cases that match the current problem. An intelligent system is also expected to behave in a consistent manner in unexpected situations, to provide solution and advice to either new or similar problems, and possibly to learn from experience as well. This sort of required behaviour justifies the need for the type of Domain Knowledge as discussed in *Section 4.1.1.1*.

Figure 4.4 illustrates the Domain Knowledge layer which is separated into the Case Library and Facts Base. The Facts Base may be represented in sentences, lists or any other formats depending on the programming environment.

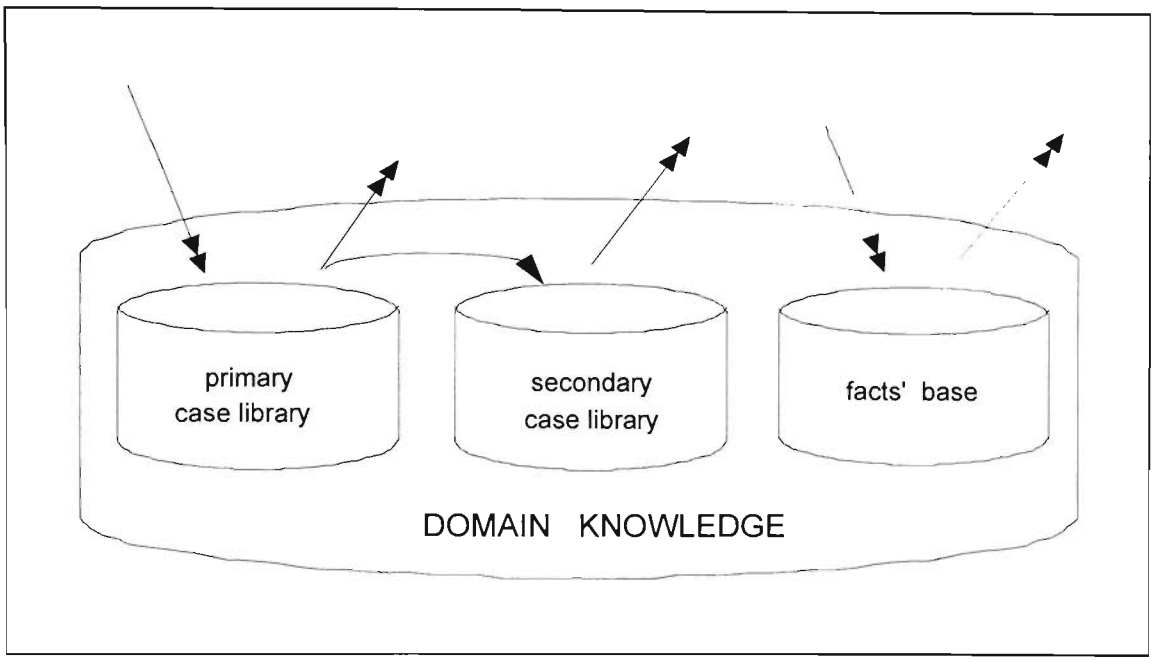


Figure 4.4 Domain Knowledge

The Domain Knowledge diagram (Figure 4.4) illustrates retrieval is first performed on the primary case library. Any similar cases found will be retrieved, else searching will proceed to the secondary case section. In other situations, for example, when searching fails to locate any similar cases, solutions could then be derived from inferencing the knowledge in the facts base.

Decomposition promotes modularity, which in turn, yields benefits such as conceptualisation and ease of maintenance. Therefore, having the knowledge base decomposed into Domain Knowledge and Inference Knowledge would yield the same returns in addition to the advantages stated in *Section 4.1.1.2.* The multireasoning paradigms representing different inference mechanisms, are employed by the system. These paradigms are organised into layers in the Inference Knowledge. Such organisation gives the system an added flexibility to switch between the reasoning paradigms whenever necessary during problem solving.

Some sort of control is required by most, if not all, systems to manage the problem solving task and coordinate the operations carried out by the different entities, modules or agents in the system. However, in some small systems, the control may be incorporated into one of the module or entity. In larger systems, it would be more efficient to have a separate control eg. a Controller agent, to manage and direct the general operations of the system. The agents in the architecture are organised hierarchically to give the Controller agent the means to regulate and schedule the system's activities.

The Interface Manager usually resides at the outer layer of the agent. In a homogeneous environment, this interface layer may be omitted as communications would be conducted in the same language. On the contrary, in a heterogeneous environment, the agents may need to interact with agents from other platforms. Therefore, an interface layer would be necessary to interpret and translate from one language to another to ensure successful communications and interactions between the different systems.

The agent architecture shown in Figure 4.5, is adopted to develop the intelligent system in power system protection.

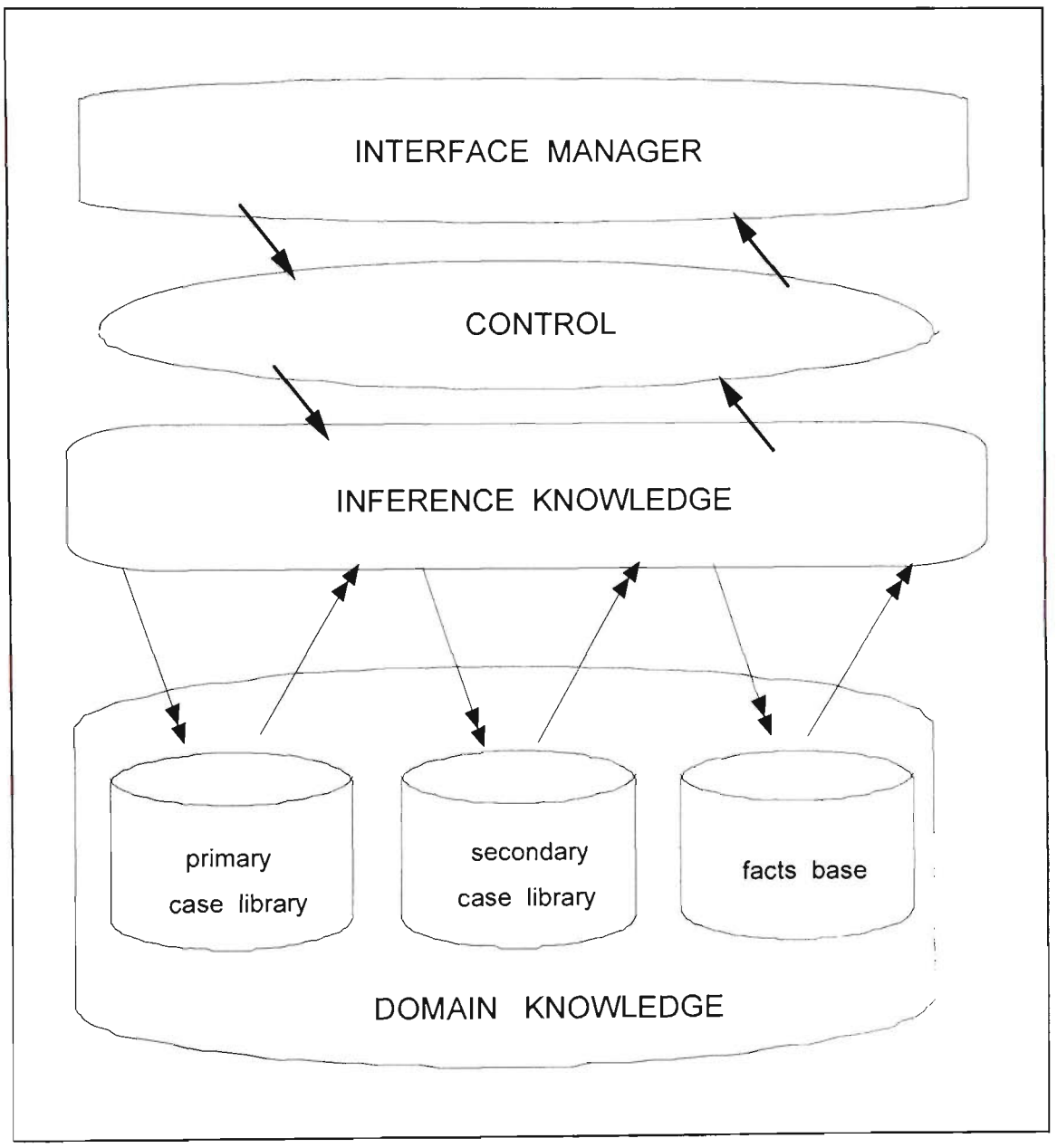


Figure 4.5 Knowledge Base and Control

4.3 Issues of Considerations - Comparing with Alternatives

The design of an architecture is like an art. Various architectures could be applied to develop the same system as there is no one sure definite design for a system. However, evaluations on the suitability of one architecture can be made by considering other alternatives.

4.3.1 Expert Systems

One of the popular systems that has gained worldwide interest in industry, government departments and academia is (rule-based) expert systems. Expert systems are computer programs that emulate the behaviour of a human expert within a specific knowledge domain. They are known to be monolithic systems with a centralised control. The main components of an expert system are the inference engine and the knowledge base. The inference engine is the control structure which allows various hypotheses to be generated and tested. It usually employs a backward and/or a forward chaining process to deduce the final solution to a given problem. The knowledge base contains a set of facts and heuristics (rules of thumb) about the problem domain.

Expert systems are also known as production-rule systems because the knowledge base usually contains a large number of formal, expert rules. The difference between intelligent systems and expert systems is the qualitative knowledge that intelligent systems possess which determines the right qualitative technique to apply to a problem with a given characteristic. One of the limitations of expert systems is that a moderate sized system could easily possess up to a few hundred rules. Moreover, they are only successful if the problem domain is well defined.

Accordingly, there are three important rules in developing expert systems which are: "*pick the right problem*", "*pick the right problem*" and "*pick the right problem*" [Liebowitz 89]. This means that choosing the right problem area is a critical decision in the development of expert systems. If the problem area is too large, the development process could be very tedious and complicated as the

knowledge base could be packed with hundreds or even up to thousands of expert rules. Otherwise, if the knowledge base is incomplete, then the answer obtained from the system will most likely be incomplete or incorrect, no matter how well the inference engine performs.

The agent could be implemented as a traditional expert system which employs rules only in its knowledge base. However, to do so would make the agent less '*confident*' in solving any problems assigned to them. The solutions derived by these agents are basically 'textbook solutions' or solutions that follow the 'rules of thumb' which still need to be verified and proven before they could be implemented in real life situation. Moreover, there are still other various limitations in expert systems which include the inflexibility of the system to changes. Any modifications to the system due to either changes in the user needs or the system requirements mean that the related rules would have to be changed or modified as well. This could result in a major change to the knowledge base which would be cumbersome and tedious to perform. The inflexibility of expert systems is one of the main reason which makes them undesirable in many problem domains where there are a lot of uncertainties existing and solutions cannot be deduced from just a simple application of a forward or backward chaining process. This is why, researchers and system developers eventually divert their attention to other type of systems.

4.3.2 Domain Knowledge

The agent architecture designed here (Figure 4.3) contains a domain knowledge which consists of a divided case library and a base of facts and

heuristics. As an alternative, similar to many other systems, there could be just one domain knowledge containing a base of facts and heuristics. While this is feasible for some systems where the accuracy of the output is not as critical as the speed, it would not be appropriate for other systems. In real time systems, speed is crucial and every second counts. Therefore, it would be desirable to reduce processes that are time consuming (e.g., searching large databases). However, in application systems where time is a secondary factor compared to the accuracy of the output, it would be beneficial to store actual cases/experiences in the case library or case memory as part of the system's domain knowledge.

Moreover, it is important for an agent to carry out its tasks in an intelligent manner, that is, having the capability of applying its experience and knowledge on what it has learned about solving a similar problem. In accomplishing this, a cost effective way of solving problems would have been achieved as well (i.e., deriving solutions through reuse of knowledge).

One of the main aims of maintaining a case library is to accumulate and preserve the knowledge of past experiences or design cases that have been tested and implemented. In addition to '*successful*' cases, a case library may contain '*failed*' cases as well [Sycara 88]. The advantage of keeping the latter is that '*failed*' cases could serve as a reminder of any unsuccessful cases so that the system could avoid repeating the same mistake and learn from experience. This behaviour imitates the human behaviour who try to avoid encountering and repeating any bad or unsuccessful episodes or events they may have experienced previously.

As discussed in earlier section, the case library is divided into two sections. The purpose of this separation is to allow modified but unverified designs to be kept in the secondary case library with the aim of enriching the system domain knowledge. If the case libraries were to be merged, then the disadvantages arising from the merger would include:

(i) larger search space is resulted, thus increasing the time to search for a particular case;
(ii) some sort of indexing technique would need to be developed to index the cases and to differentiate the verified cases from the unverified ones;
(iii) a larger memory space is required to keep the case libraries together as it would be difficult, if not impossible, to split the case library.

4.4 Conclusion

The study of architectures is not a simple task because the space of possible designs applicable to one system is unbounded. It requires the understanding of the advantages and implications of different designs. Understanding a design also means understanding how it relates to a niche and how the changes in a design affect the niche as well.

This chapter has presented a different approach to designing the architecture of an agent architecture. It uses an adaptive approach which studies the system requirements and construct the system/agent incrementally. As the components

are developed, the components are assessed before they are added to the system. Alternatives to the design are presented to enhance the understanding of the architecture. It is not easy to fully understand the underlying characteristics of a system. Therefore, the study of architectures is important to aid this understanding. Detailed presentation on the development of the agent architecture is presented and discussed in the following chapter.

Chapter Five

AGENT DESIGN AND ARCHITECTURE

Abstract

The issues relating to an agent architecture are presented in detail. The agent is constructed using a multilayered methodology. It is the ability of the agent to use multiple reasoning strategies and switch between them when necessary which reflects the human expert's way of thinking. This is central to the success of the system. Other important and related issues including layering architecture, constructing and manipulating the agent knowledge base and inference knowledge, multiparadigm approach, the hierarchical organisation of the system architecture are also discussed in depth. These issues are discussed and resolved in order to develop a good architecture for a multiagent system. However, the system architecture presented here is applicable to a wide range of decision support systems applications.

5.0 Introduction

The first generation of artificial intelligence systems were mostly monolithic, isolated and stand-alone systems. These systems are insufficient to adequately and efficiently address the complexity, diversity and performance challenges of complex, heterogeneous, large-scale applications of today systems [Vranes 95]. They are fast becoming obsolete as they are being substituted by more operational, real world knowledge based and intelligent systems to meet the complex, heterogeneous and diversified demands which represents today's real world requirements.

In fact, many of these knowledge based systems are also intelligent systems. They normally employ distributed problem solving together with various AI techniques to enhance the system skills and expertise in solving problems. According to Tenney and Sandell [Tenney and Sandell 81] :

> *"Broadly speaking, the basic problem motivating a study of distributed decision systems is a desire to control a system which is significantly more complex than any single agent can deal with alone."*

Lately, agent technology is being applied widely in many diversified application domains such as medical, industrial, engineering, design in order to capture and simulate the human expert's way of thinking and handling any problems [Huang *et al.* 94, Hayes-Roth *et al.* 94]. This is accomplished by incorporating and encapsulating cognitive science such as behaviour into the agent.

The study of software agents has resulted in a diverse set of views and realisations [Riecken 94]. According to Riecken, these views include:

- building a specialised agent to assist a user to perform a specific task (e.g., scheduling an itinerary or ranking and presenting e-mail and news;
- integrating the performance of sets of the specialised agents (e.g., several agents get together to schedule a meeting;

- integrating agents to create an "assistant" (e.g., a software assistant who recognise, classify, index, store, retrieve, explain, and present information relating to human-computer interactions).

As reported by Isbister and Layton [Isbister and Layton 94], several analysts believed the future of computing lies in communication, and therefore the need for filtering. Agent software may be able to do some of the routine monitoring tasks people now do by hand that do not require sophisticated judgement [Isbister and Layton 94]. It may also provide a place for people to deposit and gather annotations about information sources.

Agents are being in used for many reasons. Some of these reasons are [Isbister and Layton 94] :

- agents can provide assistance to users in offering advice, suggestions, training, etc.;
- agents can make users more productive/effective if implemented to address the needs of the users by allowing users to focus on critical tasks;
- there are lots of cpu cycles that go unused that could be working for the user while they are busy on other tasks;
- users can off-load repetitive and/or mundane tasks to their agent;
- agents can be equipped with knowledge of things that users should not need to know (e.g., location of files or knowledge about the network);
- agents provide a good metaphor to help deal with delegation and communication in the interface.

Agents have also been in used in many different situations and in various diversified areas. One of the most popular application is using agents as interface agent which aimed to provide an intelligent and friendly user interface system [Maes 94, Lashkari *et al.* 94].

This chapter discusses the development of the generic agent architecture using the Adaptive Approach presented in the previous chapter. The design of intelligent agents is an important research direction within the field of multiagent systems [Bond and Gasser 88, Durfee and Rosechein 94], where the behaviour of a society of agents is described by modelling the individuals and their interactions from an agent-based perspective. Finding appropriate architectures for individual agents is one of the fundamental research issues in agent design. There are two major reasons for dealing with agent architectures. One reason is to explain and to predict the agent's behaviour and the second reason involves the design of multiagent systems.

5.1 Agent Design

An agent is a complex artificial intelligence system which possesses substantial knowledge and reasoning components. The agents are constructed using a number of distinct and loosely coupled layers. Layering is a powerful technique for the design of resource-bounded agents as it combines a modular structure with a clear control methodology. It supports modelling of different levels of abstraction, reasoning and complexity of knowledge representation. Some criticism has recently been levelled at certain aspects of layered architectures [Wooldridge and Jennings 94]. However, the layered methodology

offers a more natural, flexible and versatile approach for modelling intelligent systems.

The agent architecture is shown in Figure 5.1.

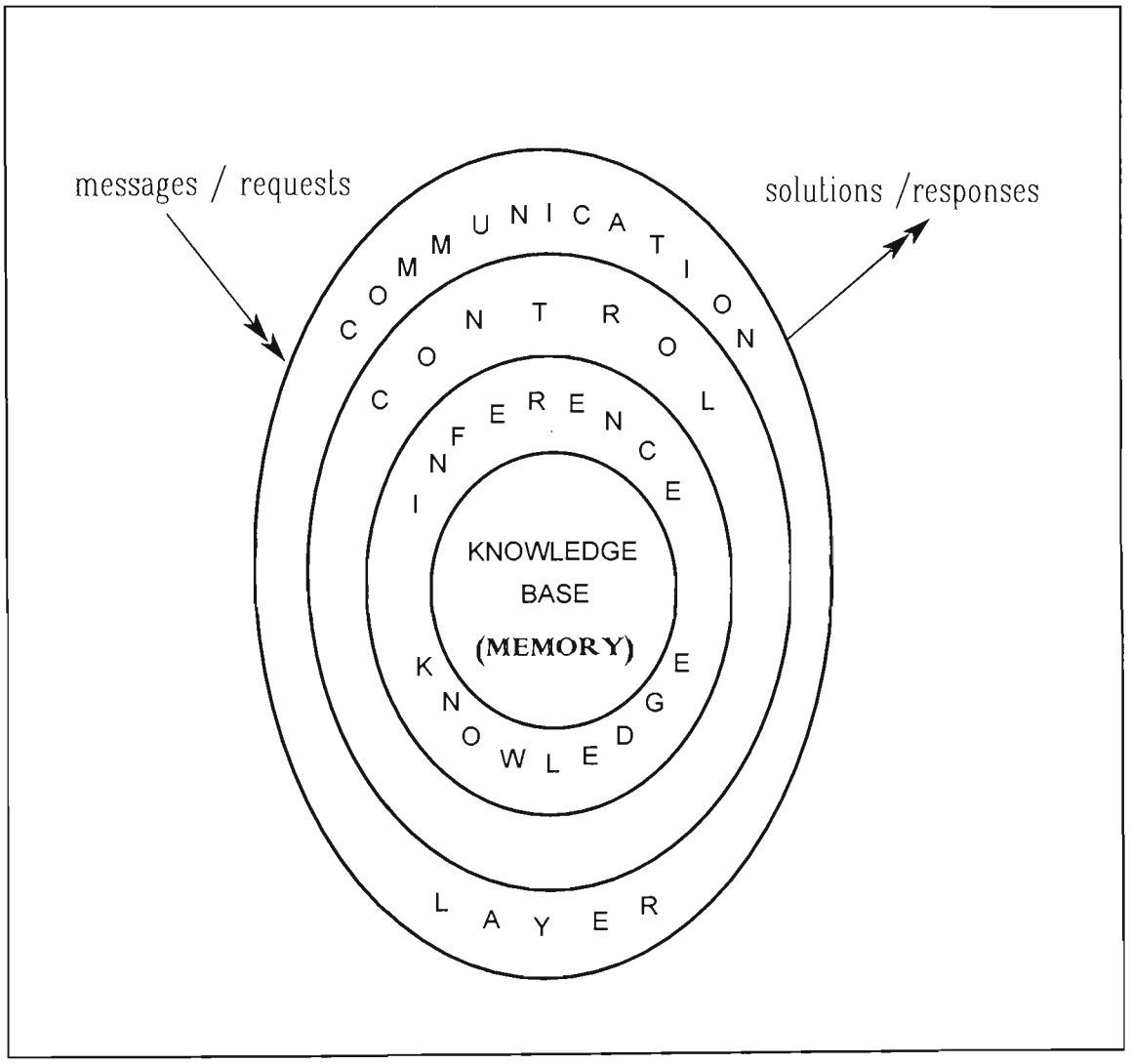


Figure 5.1 Agent Architecture

Each agent consists of a multilayered structure containing sublayers within layers. The top layers include :

- Communication Layer[1]
- Control
- Inference Knowledge
- Memory or Knowledge Base

The architecture uses a multiparadigm approach leading to the development of an agent which resembles the human expert's way of thinking. This gives the agent the ability to use multiple reasoning strategies and switch between them whenever necessary. This feature contributes largely to the success of the system.

5.2 Agent Architecture

The agent behaves similarly to the way a human expert might behave when solving a problem. The agent architecture comprises a communication layer, a control, a working memory and multiple layered knowledge. The *memory* consists of the domain knowledge which is divided into two sections - facts base and case library.

Figure 5.2 shows a more complete diagram of the layered agent architecture with the inclusion of a selector located at the top of the inference layer and the separation of memory into case library and domain knowledge.

[1] In some systems where the development and interoperation between different systems are performed in a homogeneous platform and communication is conducted in the same language, no translation is then required. As such, the communication layer may become unnecessary and therefore, could be omitted in the agent design.

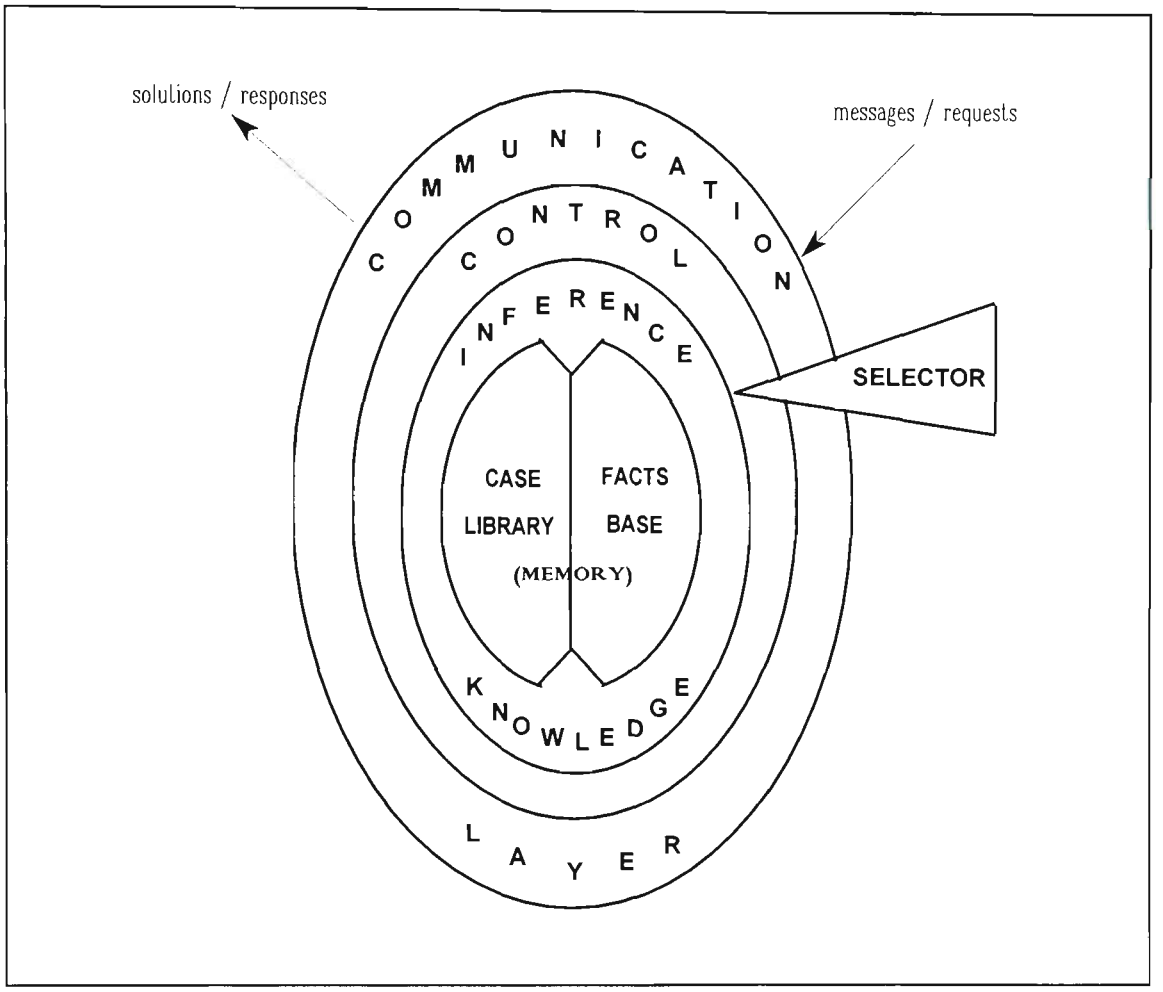


Figure 5.2 Layered Agent Architecture

As shown in Figure 5.2, the agent's architecture is organised into several layers. The multilayering architecture demonstrates a powerful way of organising the components within the agent. The different sublayers have different responsibilities and the interactions between the layers are achieved using message passing. The layers in this architecture do not communicate with one another directly but through the `Control`, unlike the layering architecture proposed by Hayes-Roth *et al.* [Hayes-Roth *et al.* 94] where adjacent layers can communicate with one another. And within the layers in the Inference Knowledge, a selector is employed. The purpose of the selector is to trigger and manage the execution of the correct inference mechanism during problem

solving. The reason for not allowing direct communication between the layers is to reduce the traffic flows between the layers.

The `Communication Layer` is the interface layer where the agent communicates with the outside world, that is, with a human or another agent. This layer sends messages to and receives messages from external sources. Messages received are translated if necessary, and passed on to the `Control` for further actions to be taken. Based on the messages, the `Control` may decompose a problem, schedules the tasks and pass them to the `Inference Knowledge` via message passing. The `Inference Knowledge` invokes and utilises the appropriate inference mechanism(s) through the `selector` which is kept in *memory* to carry out the assigned tasks. The results or solutions to the tasks are then returned to the `Control`, and are communicated back to the external sources through the `Communication Layer`.

For example, when a message requesting a solution to a problem arrives, the `Communication Layer` will interpret it as a request which requires deliberative action/response. It may be required to translate the message before passing it on to the `Control`. The `Control` then attempts to decompose the problem into smaller and more manageable tasks. It then calls on the `Inference Knowledge` to perform these tasks. The selector in the `Inference Knowledge` then triggers and executes the appropriate inference mechanism depending on the current state of problem resolution. When a final solution is eventually constructed or derived, it is then conveyed back to the `Control` which is then conveyed back to the external source. Further actions may be required (e.g., to explain or justify the solution reached, which would trigger a similar communication process).

As noted in the previous diagram, the control in the system is built as a distinct layer explicitly. Such explicit control provides the advantages of modularity and flexibility to the problem solving system [Mookerjee and Chaturvedi 93]. In addition, when a change to the problem solving strategy system is introduced, it does not require any change in the domain knowledge at all - only the control knowledge needs to be changed or updated. Furthermore, with the clear separation of the control and domain knowledge, heuristics or any other strategies could be introduced, added, deleted or modified, without affecting the domain knowledge. Another additional advantage of this arrangement as will be demonstrated in later chapters, is that it allows the system to be adaptive as well as to respond dynamically to changes in the user objectives.

5.2.1 The Layers Within the Agent

The `Communication Layer` interprets the messages received and the `Control` responds them appropriately to these messages by taking necessary actions. To be able to function effectively and efficiently on its own, the agent needs to be able to react and respond correctly to all messages it may receive. The responses may require *deliberative* or *reactive* actions [Huang *et al.* 94]. *Deliberative* actions are actions taken deliberately in response to a request or a message, for example, plan selection, task decomposition, task allocation and scheduling. *Reactive* actions are concerned with the agent's timely responses to another action (e.g., the arrival of new data or changes in existing data). The `Inference Knowledge` forms the central component of the agent

architecture, which enables the agent to derive conclusions using the facts and knowledge stored in the Knowledge Base or Memory.

5.2.1.1 The Communication Layer

The Communication Layer resides at the outermost layer of the agent architecture and acts as the interface layer. It interfaces and interacts with the entities of the outside world. These entities could be a human beings or other computer systems. In other words, the Communication Layer is responsible for communicating with other internal or external agents by sending messages to and receiving messages from other agents.

In order to be able to carry out its responsibilities, the communication layer should possess the capability of being '*multilingual*'. This is even more true if the agent needs to interact with other agents or entities in a heterogeneous environment. The interpretation and translation of messages from one language to another are performed in this layer. Therefore, this layer could be omitted if the agents are operating within a homogeneous environment.

5.2.1.2 The Control

The Control lies at the next layer below the Communication Layer. The Control is basically responsible for controlling the activities of the agent. All messages/problems received from and all responses/solutions relayed back to the outside world are transmitted through the Communication Layer.

The Control performs task decomposition upon the receipt of a problem from the outside world. In other words, the Control manages and schedules any task that comes into the system. The Control also directs, controls and coordinates the overall functionality and operation of the system. Furthermore, the Control also schedules the agent activities and passes the decomposed tasks to the Inference Knowledge which is situated at the next lower layer.

5.2.1.3 The Inference Knowledge

The next layer lying below the Control layer is the Inference Knowledge. The Inference Knowledge specifies the basic inferences that can be used and applied to the Domain Knowledge. The different reasoning mechanisms are organised in layers within the Inference Knowledge. These include heuristics, analogical reasoning, case based, explanation based and many other reasonings paradigms. Residing on the top of these layers is a selector.

The selector residing inside the inference layer is capable of selecting the most appropriate reasoning paradigm or may trigger a number of paradigms simultaneously. The selector is responsible for the appropriate reasoner to solve a given problem. In other words, the selector is responsible for the automatic switching between the different reasoning or inference mechanisms, when solving the problem.

When the Inference Knowledge receives a task, the selector attempts to solve it by applying the knowledge kept in the *memory*. The selector may trigger different reasoning paradigms where appropriate such as case based, rule based, explanation based or argumentation to deduce an appropriate solution to the assigned task. When solving a task, the selected reasoning paradigm employed or selected depends on the task and on the eventual results. For instance, if case based reasoning fails, rule based reasoning is then used to deduce the solution.

5.2.1.4 The Memory - Domain Knowledge

The Domain Knowledge stores past experiences and facts about the domain area and is divided into two sublayers - facts base and case library. The organisation of these sublayers is shown in Figure 5.3.

The Domain Knowledge layer can be viewed as just a collection of facts and knowledge about the application domain. However, the domain knowledge alone does not constitute an agent intelligent. Therefore to '*inject*' intelligence into the agent and to allow the agent to achieve its goals, the multiparadigm inference mechanisms is employed.

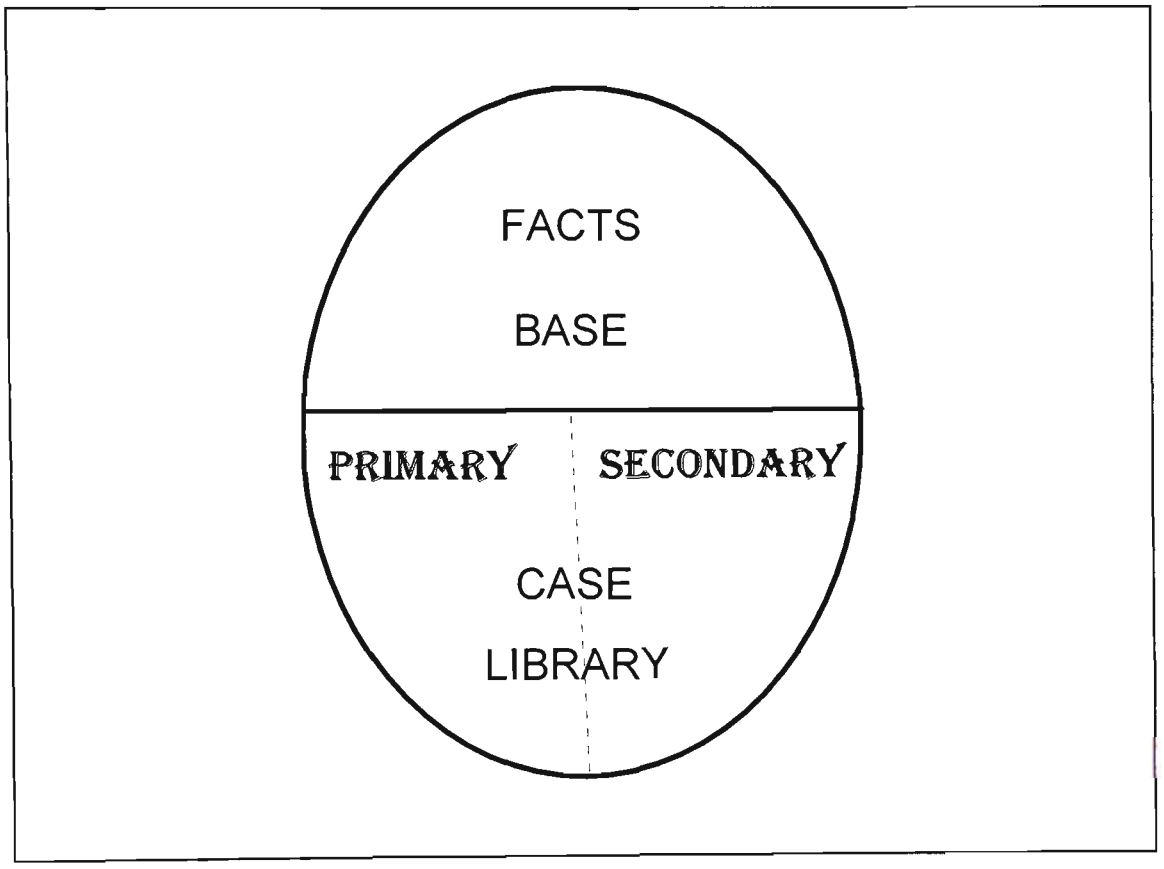


Figure 5.3 Agent's Domain Knowledge

An added advantage of separating the factual knowledge and the case library is that the system will be more reliable. Reliability is a very important factor in some systems. This is even more true in application domains where the accuracy of the system's output is vital, and any defective or inaccurate output could bring disastrous consequences. Therefore, when a solution is retrieved from the primary memory, it guarantees that the solution has been verified by an expert or proven to work in practice. Otherwise, if the solution is retrieved from the secondary memory or derived by other means, a word of caution should be given to the user about the unverified solution.

5.2.1.4.1 Case Library

The `case library` represents the agent's expertise in solving problems. The `case library` layer contains facts about the application domain, which are better known as cases. These cases represent the actual design problems solved by the agent previously. This section of the *memory* contains past experiences or cases that have been previously solved by the agent. Not all cases are stored in memory. Only new cases which are different from the existing cases in the `case library` are kept. Furthermore, the `case library` is divided into two parts - the primary and the secondary.

The primary section of the `case library` stores the cases which have either been proven by actual successful implementation or verified by a human expert. Whereas, the secondary section keeps the newly constructed cases as well as those cases which have been adapted and modified but have not been proven or verified yet. Retrieval is mainly carried out from the primary section only. In situations where there is no similar cases found, retrieval is then performed on the secondary section. However, designs in the secondary section could be promoted to the primary section when the designs have been verified by an expert and tested to work in practice. This separation and organisation of the `case library` provides a more efficient retrieval and searching of similar cases.

5.2.1.4.2 Facts Base

The `facts base` contains all the rules and facts about the domain, but it provides no information about how the knowledge should be utilised. Example of some of the facts in the `facts base` are the current transformers requirements for a certain protection scheme (e.g., *a high impedance scheme requires dedicated current transformers with equal turns ratio)*, detailed information on available relays and so on.

The `facts base` layer contains raw facts about the application domain. These facts may be represented in a number of different forms, depending on the programming environment. For example, when using a knowledge base environment, they could be represented in the form of lists.

5.3 Multiparadigm Approach

History has shown that traditional expert systems have experienced a lot of limitations and drawbacks [Wielinga *et al.* 92]. In a bid to avoid the unsatisfactory situations experienced by the traditional expert systems, a large number of methodologies and AI techniques were introduced (e.g., fuzzy logic, genetic algorithms, causal reasoning and case based reasoning). On top of that, multiparadigm approach was also introduced. This approach attempts to take advantage of the efficiency and competency of each paradigm by integrating them into a consistent and coordinated strategy.

Multiparadigm approach is becoming more popular in today's systems [Vranes 95, Hayes-Roth 94]. It increases the robustness, efficiency, capability and intelligence of the system; enabling the system to handle more varied, complex and new problems. This is contrary to the traditional expert systems which mainly consist of rules in the knowledge base and which use forward or backward chaining process as the inference engine. These type of systems are known to be rigid and brittle. They have limited capability and their efficiency in problem solving are restricted to what have been defined in their knowledge base. They seldom employ any learning mechanism and are usually unable to solve any problems beyond their defined but limited knowledge.

Each paradigm, whether it is reasoning or knowledge representation has its own merits and advantages, and offers certain benefits in comparison to other paradigms. However, to rely on a single paradigm only would result in the system which is too rigid or inflexible and has a narrow focus. Also, it becomes difficult, if not impossible, to describe all aspects of a large complex system. Hence, a multiparadigm approach is adopted because the system can benefit from the utilisation of as many paradigms as it needs - each paradigm handles one aspect of the system that it is best suited. In addition, the different paradigms or mechanisms employed can be used to complement one another.

5.3.1 Case based reasoning

One of the main reasoning paradigm employed by the agent system is case based reasoning[2]. Case based reasoning (CBR) is chosen because it fits into the

[2] Refer to *Appendix F* for more information.

application domain naturally. Case based reasoning is a technique that builds a case library with new cases and problem solving is achieved by retrieving similar cases and adapting the solutions to the problem case. According to Aamodt and Plaza [Aamodt and Plaza 94], case based reasoning is in effect, a cyclic and integrated process of solving problems, learning from this experience, solving new problems again, and so on.

Case based reasoning is used by humans extensively at all times [Kolodner 93]. For example, attorneys are taught to use cases as precedents to construct and justify their arguments. Other professionals that use the same reasoning technique include architects, mediators, arbitrators and general practitioners. In fact, we all do similar things- (e.g., in planning our housework and officework activities). We remember what worked and what did not work previously and based on our experience, we follow old plans or create new plans to perform our task. Case based reasoning is a simple and yet powerful technique which is easily suited and applicable to many problem domains. It is a popular technique because of its simplicity, and its natural ability to reason that mimics the way human would normally go about in solving a problem. This paradigm is given more consideration, and is further studied and investigated. This investigation resulted in full elaboration of the paradigm where the issues concerning case based reasoning is discussed, the suitability, the advantages and the limitations of case based reasoning are explored.

The representation of encapsulated case enables easy storage and retrieval of information and data. In practical situations, solving a design task or any problem for that matter, never or very rarely begins from first principle or from scratch. People always refer to their experiences in dealing with similar tasks

before they start to work on any new problems. This is the general human approach to problem solving. The fact that case based technique resembles very closely the natural way of reasoning that people employ in their daily decision making process makes it even more appealing and appropriate to use.

Past experiences or past designs which represent the domain knowledge are encapsulated as cases. A case contains all related information encasing it which may be stored as a self-contained case or it could be broken down to smaller units. These sub-units may be kept in a case and linked to one another via pointers. The advantage of this method of storing cases also ensures easy retrieval of information and data.

A typical case based reasoner usually starts with a representative set of cases which covers the goals and sub-goals that may arise in reasoning [Kolodner 93]. It includes both the successful and failed attempts at achieving those goals. The purpose of keeping the latter cases is that these cases will serve as a reminder of unsuccessful past experiences and how the user could refrain from repeating the same mistakes again. As the agent also employs other reasoning techniques to supplement case based reasoning, it does not necessarily have to start with a representative set of built-in cases. However, it is usually preferable to '*initialise*' the system with a predefined set of qualified cases in the case library.

There are several principles governing a case [Kolodner 93]:

- A case represents specific knowledge tied to a context. It records knowledge at an operational level.

- Cases can come in many different shapes and sizes, covering large or small time slices, associating solutions with problems, outcomes with situations, or both.
- A case records experiences that are different from what is expected. However, not all differences are recorded, only cases that could teach a useful lesson would be '*worthy*' of recording as cases.
- Useful lessons are those that have the potential to help a reasoner achieve a goal or set of goals more easily in the future or that warn about the possibility of a failure or point out an unforseen problem.

So, what is a case, exactly? As defined by Kolodner [Kolodner 93]:

> *"A case is a contextualised piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner."*

According to Aamodt and Plaza [Aamodt and Plaza 94], a case in CBR terminology, usually denotes a problem situation. A previously experienced situation, which has been captured and learned, and which can be reused in solving future problems is referred to as a past case, previous case, stored case, or retained case. Correspondingly, the description of a new problem to be solved is termed as a new case or unsolved case.

When the agent attempts to solve a new problem, it first applies case based reasoning. This is done by looking into its case library or case memory in search of a case which has similar attributes or characteristics to the current problem. If such a case exists, the case is retrieved. The solution of the retrieved case is then

adapted to the current problem if necessary to form a new case solution. However, if no similar case is found, or the adapted solutions not acceptable, the agent then applies another reasoning technique to solve the current problem.

5.3.2 Rule based reasoning

There has been a major shift over from production rule systems to AI systems which utilise AI techniques rather than just rules in problem solving. It is indeed one of the consequences when production rule systems have been exposed of their limitations. This is largely due to the fact that a moderately sized system could easily contain up to a few hundred rules in the knowledge base. A change in the system or user requirements may cause a chain of modifications in rules. Thus, production rule systems are known to be very brittle and not susceptible to any changes once the system has been implemented. However, even though it suffers from a number of limitations, rule based reasoning is still very popular because it is easy to use and implement.

Rule based reasoning is one of the simplest and most common reasoning techniques that has been used in many systems ever since the early days of expert systems. It takes the form of '*if-then*' construction which is a straight forward reasoning technique and is mainly employed in the development of expert system applications. Rules are most efficient and can be just as powerful in systems where the problem domain is well-defined and a solution can be derived using a forward or backward chaining process.

The agent employs rules as a secondary reasoner. Case based reasoning is said to have failed if there is no similar cases retrieved from the case memory, or the user rejects the adapted solution. In these situations, the agent uses rules as an alternative reasoning technique and applies the facts in the domain knowledge to deduce a new solution. The result is a new solution case which is then saved into the case memory.

5.3.3 Argumentation

In addition to rules, argumentation is also employed as a secondary reasoner. Argumentation is a decision procedure based on a simple flexible method of reasoning under uncertainties for argument generation and aggregation [Huang *et al.* 94]. Decision making is often complicated by the presence of incomplete or even conflicting information. For example, a student who just finished high school has several options to take. The student could study further and take up a degree course in a university or a short course in a local institution of higher education. Alternatively, the student could decide to take up apprenticeship or start looking for a job. These are some of the many course of actions that are available to the student. In making a decision, the student has to consider the pros and cons of each action. Moreover, he or she may have incomplete or conflicting information but will still need to make a decision using the knowledge at hand.

The agent operates in a similar fashion. For example, given a problem or task to be solved, there could be a number of alternatives or courses of action that are applicable and could be taken. Say, for example in power system

protection where the problem is to design a protection scheme to protect a busbar of a power system. In addition, suppose that high impedance differential scheme or medium impedance differential scheme could be applied. Alternatively, overcurrent scheme could also be used. However, if high impedance differential scheme were to be applied, dedicated current transformers (cts) must be made available. Other requirements on the cts are that they must be sensitive enough to saturate under all through fault conditions; they must be of the same turns ratio and also of low reactance type. On the other hand, medium impedance differential scheme, which is just as efficient as high impedance differential scheme could be applied as well. However, this scheme may be too expensive, but it is not as demanding on the cts characteristics and dedicated cts are not necessary. At the same time, various ratios of the main cts and their other load than that of the differential scheme are acceptable too. Both, the high and medium differential impedance schemes, are known as unit schemes where backup protection for the downstream component is not possible. As a comparison, the overcurrent protection scheme would be less efficient (i.e., slow in operation) but it would cover all types of faults on the busbar and backup protection could be arranged. Therefore, to facilitate decision making in such a context, a domain independent decision procedure must be abstracted and constructed [Huang *et al.* 94]. This procedure is separated from the domain specific knowledge which also permits the formalisation of decision knowledge.

There is always a goal to be achieved in the process of decision making, represented as a *decision context*, which could be specified by the user or generated by the system in operation. Referring to the example in the previous paragraph, the goal could be to design a protection scheme which has to be fast

and efficient, and the incurred cost must be taken into consideration. Provision for backup protection is not important. In this context, there are advantages and limitations for each proposed option. This eventually would be combined to give a most preferred decision (i.e., to employ high impedance differential scheme, assuming that dedicated cts are available).

5.3.4 Explanation based reasoning

Explanation based reasoning is modelled after an explanation based decision making process through which people are believed to perform some decision making tasks [Hair *et al.* 92]. Alternative explanations can account for a given set of data and the eventual decision taken depends on the strength of each explanation. An explanation is a causal model which incorporates all available data into a coherent structure that supports one of the possible decision outcomes. Hence, explanation based reasoning would be suitable in situations where explanation is required for a course of action or decision taken for a given set of data.

An interesting observation was made by Minton *et al.* [Minton *et al.* 90] :-

> *"..... as people use experience to do planning and problem solving, they often use remindings of old experiences as the starting point for explanation."*

On this basis, explanation based reasoning is employed by the agent as the explainer to give the user a satisfactory explanation as to why a particular

protection scheme is proposed for a specific power system or a part of it. For instance, when a solution is recommended or proposed, the user may enquire as to how the solution has been derived.

5.4 The Agents' Function

There are basically three types of agents which have been modelled and developed from the agent architecture presented in the previous section. These agents are simply known as the Interface, Coordinator and Design agents. Each of the agents has a distinct and separate responsibility to perform in a multiagent system. They are known as the semi autonomous agents whereby each agent has sufficient knowledge and expertise to work independently to generate partial solutions only. Hence, they need to cooperate and communicate via message passing or blackboard to formulate an integrated, consistent and complete solution to the original problem.

5.4.1 The Interface Agent

The purpose of having an Interface agent is to relieve other agents of the responsibility to interact with the user or an external system. In a heterogeneous environment, the Interface agent conveys the user's request or information to the appropriate agent and relates the agent's response or solution back to the user. The Interface agent may be required to interpret or translate the messages that are passed between the system and the user.

However, not all systems require an Interface agent, for example, small systems or systems which contain a society of autonomous or self governing agents. In the former systems, it may not be feasible to employ and maintain a separate Interface agent. In the latter case, the autonomous agents are normally able to carry out the job by themselves or with some assistance from the neighbouring agents. In both situations, the agents would be equipped with the capability to interact and communicate with one another or with the user directly. However, on the issue of system design, it would be much neater and better - conceptually and logically as well, to have an Interface agent to specifically handle all the communications and interactions with the outside world.

5.4.2 The Coordinator Agent

There is usually some sort of control defined in every system - be it distributed or centralised. In a hierarchical organisation, the control is usually held by one entity which controls and coordinates the activities in the system. Here, a Coordinator agent is introduced to carry out the coordinating and controlling activities. The extent of control possessed by the Coordinator agent depends on the system requirements and the design of the system architecture. That is, the control may be shared with other agents in the system or the Coordinator agent may have complete and total control of the system.

The Coordinator agent may not need to have the total control of the system. Rather, it could take on the responsibility of overseeing the entire system, and decomposing an original problem into smaller tasks which are then distributed

to the appropriate problem solving agents. However, one of its main responsibilities of the Coordinator agent is to coordinate partial solutions generated by the problem solving agents into a complete and coherent solution.

5.4.3 The Design Agent

The Design agents are sometimes known as the problem solving agents. Each of the agent has incomplete knowledge of the Universe of Discourse[3] but is an expert in a particular area of the domain. Therefore, each agent is responsible for delivering a solution or a decision which forms a partial solution to the whole problem. These partial solutions are then sent to the Coordinator agent for coordination.

The Design agents need not to communicate with one another if there is no need for it. However, they have the same responsibility - to provide solutions to the tasks which have been assigned to them. Tasks are assigned to them by the Coordinator agent which expects solutions to be delivered back for further coordination. The communication can be carried out via message passing or with the aid of a blackboard system. The Design agents may also communicate with the Interface agent when they require further information from the external source about the tasks they are currently solving.

[3]Universe of Discourse (UoD) is also referred here as the problem domain modelled by the system.

5.5 Conclusion

The agent architecture developed using the multilayered and multireasoning paradigm add to the merits of reusability and the agent's intelligence level. Some of the advantages of the agent architecture presented include :

- The knowledge base is divided into two layers which are populated by the `Domain Knowledge` and `Inference Knowledge`. The purpose of this functional separation of the knowledge base is to give each layer more flexibility to adapt, if necessary and to enable proper management of the sublayers within each layer.

- The purpose of employing a `selector` in the `inference knowledge` is to act as the interface between the `control` and the `inference knowledge`. The selector complement the `control` layer can be regarded as analogous to a sensor which detects the current state of problem solving and then activates certain procedures or reasoning paradigms.

- The division of the case library into primary and secondary has the advantage of generating more reliable solutions, that is, provided that the solution can be retrieved from the primary section. Otherwise, a word of caution or a warning note would be given to the user that the solution generated has not been verified.

In addition to the above, the agent architecture also leads to the development of intelligent agents which simulates the natural approach or process that the human expert employs in decision making and problem solving.

Furthermore, it is important to note that it is also possible for the agents to be organised in a variety of ways to form different system architectures depending on the environment, operating system, problem domain and constraints. This is true because there is no single design which could be used in different application domains. The following chapter illustrates the different system architectures that could be designed and built by reorganising and redefining the responsibilities of the agents in the system.

PART III

THE SOLUTION

Application and Implementation of Different System Architectures

Chapter Six

ORGANISATION OF AGENTS

ABSTRACT

This chapter examines how the system architectures change with the decreased in the autonomy of the agents. It explores the application of the intelligent agents and how the organisation of the agents could produce different possible system architecture designs depending on different factors and constraints. The different architecture designs presented are adopted from the architecture design presented in chapters four and five. The agents are represented as sophisticated expert systems that communicate their design information and knowledge using various methods of communication including message passing and use of blackboard system. These three system architectures are: (i) distributed knowledge system in a knowledge base environment; (ii) distributed/shared knowledge system in an object oriented environment; (iii) distributed knowledge within a federated system framework in an object oriented environment. The three system architectures have a hierarchical organisation because such organisation supports the natural grouping of functionally related agents to facilitate cooperative problem solving. The architecture has been designed to support task decomposition which is carried out by the coordinating agent with actual task execution performed by the design agents. A prototype system for each of the system architectures, have been built and an illustration of each system operation is also given.

6.0 Introduction

As the general design of the agent architecture has been detailed in chapter five, this chapter will focus on the organisation of the different type of agents

and their respective responsibilities at the system level. The system architecture consists of a number of distributed expert systems, also known as intelligent agents, organised in a hierarchical structure. The hierarchical organisation supports the natural grouping of functionally related agents to facilitate cooperative problem solving.

The performance of a system depends on a number of varying factors. One of the main determinant factors is the architecture of a system. Therefore, attention should be given to the study and design of the system architecture. In a cooperative system, especially in an agent based system, cooperation between agents which are also known as problem solvers, must be structured as a series of carefully planned exchanges of information. It is only recently that research are giving more considerations and placing more emphasis on frameworks and strategies for cooperation.

There are three types of agents in the system: coordinating agent, design agent and interface agent. Each agent maintains its own knowledge and an inference engine. The knowledge possessed by each individual agent is a subset of the problem domain and the agent is an expert in that particular field of the domain it represents. The agent's knowledge is represented in different forms and they employ multiparadigm reasoning strategy to improve their skills in problem solving. The reasoning paradigms used may include case based, rule based, explanation based, causal reasoning, fuzzy logic, inductive reasoning, argumentation (reasoning under uncertainties) and pattern matching.

The agent architecture is applicable for the construction of a wide range of intelligent or decision support system applications. The architecture has been

designed to support task decomposition which is carried out by the coordinating agent with actual task execution performed by the design agents. The interface agent is introduced to carry out all interactions between the system and the user.

6.1 Multiagent Systems

The agents architectures developed in chapters four and five provide the basic building blocks for the design of an agent based system. The actual system architecture will depend on a number of parameters such as application area, system requirements and environmental issues. Consequently, the systems developed differ in the organisation and management of the system operation, the responsibilities of the various type of agents, the employment of other modules or additional agents in the system and most important of all, the degree of autonomy of the agents. The following sections illustrate three system architectures for the development of an intelligent system which can be applied to a wide range of applications.

Multiagent systems are best suited for applications where *interoperability* between application programs is required. The term *interoperation* between application programs means the exchange of information and services with other programs to solve problems that could not be solved alone [Genesereth and Ketchpel 94]. Multiagent systems are thus suitable for applications where task decomposition and coordination of solutions are necessary. In order to cooperate, the agents must be able to communicate; without communication there can be no cooperation. In fact, communication is the most fundamental and important issue in system design. This is even more prominent in

multiagent systems because cooperation, collaboration, conflict resolution and control can only be achieved successfully and efficiently through effective communication between parties concerned. The two most widely used methods of communication are through message passing and the use of a blackboard[1] which is known as a global database area.

The following are the three multiagent systems that have been designed, and a prototype for each has been developed and implemented. The three systems are depicted in the order of the autonomy of the agent which decreases with each of the following system:

i) a distributed knowledge base system in a knowledge base environment [Wong *et al.* 96, Wong and Kalam 97a, Wong and Kalam 97b];
ii) a distributed and shared knowledge base system in an object oriented environment [Wong and Kalam 95];
iii) a distributed knowledge base system within a federated system in an object oriented environment [Wong *et al.* 95].

6.1.1 Agents Autonomy

There are basically three types of agents architecture as mentioned before in the previous chapter - autonomous agents, semi-autonomous agents and agents which relinquish their autonomy to another higher representative agent.

[1]The description of a blackboard as given by Nii consists of three components: knowledge sources, blackboard data structure and control [Nii 86].

With the autonomous agents, the knowledge possessed by each agent may be duplicated in all the agents. In addition to that, they may spend most of their time communicating with one another rather than performing some productive work. At the other end, there are the agents that surrender their autonomy to a representative agent. However, due to the reason that the agents do not communicate with one another directly, there may be too much coordinating work for the representative agent. The semi-autonomous agents lie somewhere in between the two extreme type of agents. While the semi-autonomous agents requires to interact and coordinate to solve a global problem, they are also capable of solving smaller units of decomposed problems.

Agents of different autonomy levels have different needs and specifications on the system requirements, level of knowledge to be maintained and so on. They have influence on the way a system's architecture is designed. Therefore, it is important to choose the right type of agent architecture for an application system.

The multiagent system consists of three types of agents which are organised hierarchically: Interface agent, Coordinator agent and Design agents. In this thesis, these agents are regarded as semi-autonomous agents. Each agent has a distinct set of responsibilities and is able to perform its task independently. However, they need to cooperate in order to integrate their solution into a coherent solution.

Further discussion of agent cooperation in multiagent systems may be found in [Bond and Gasser 88, Gasser and Huhn 87, Huhn 87].

6.2 Design of a Distributed Knowledge Base System

In this system, a distributed knowledge base system with a centralised control in a knowledge base environment is modelled and designed. The system architecture consists of multiple agents. The agents have specific tasks such as interfacing and communicating with external systems, solving domain problems, and coordinating partial solutions into a complete integrated solution. Hence, three main types of agents in the system have been identified: Interface agent, Design agents (also known as problem solving agent) and Coordinator agent. Even though the agents have distinct responsibilities, they originate from the same agent architecture.

6.2.1 The Organisation of the System

The system organisation of the agents is quite similar to the organisation of the components in an agent architecture. The agents in the system are organised hierarchically where the Interface agent resides on the boundary of the system. Due to the controlling nature and duties of the Coordinator agent, it resides on top of the Design agents. The Design agents themselves are organised in a horizontal position - as their responsibilities and their status in the system are the same.

In this system, the overall problem to be solved is decomposed into sub-problems by the Coordinator agent. This sub-problems are assigned to the appropriate Design agent. Each Design agent would asynchronously plan its own actions. The results of their actions are then returned back to the

Coordinator agent to be synthesised into a complete coordinated solution to the original problem. This problem solving process is used in the distributed problem solving system [Shaw and Fox 93]. According to Shaw and Fox, this common strategy involves four steps: problem decomposition, task assignment, local problem solving and solution synthesis.

The hierarchical strata organisation of the system agents are shown in Figure 6.1 while Figure 6.2 shows the flow of interactions between the agents.

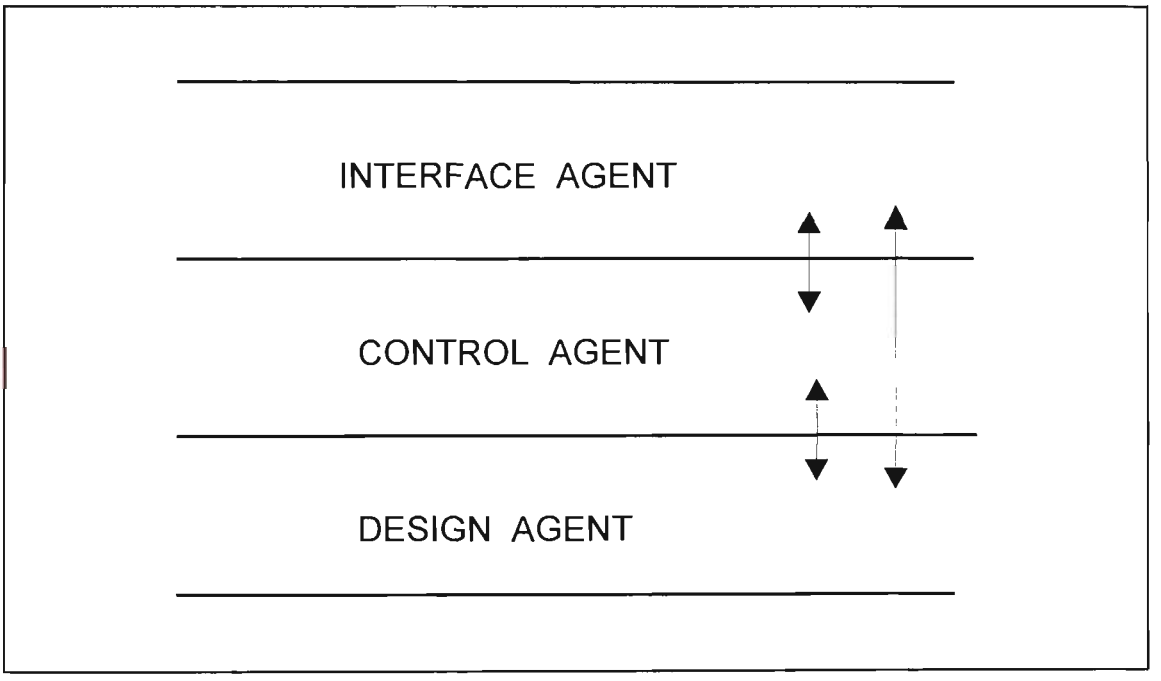


Figure 6.1 Hierarchical Strata Organisation of the System Agents

Figure 6.2 shows the hierarchical structure of the agents in the system and the flow of communication between the agents.

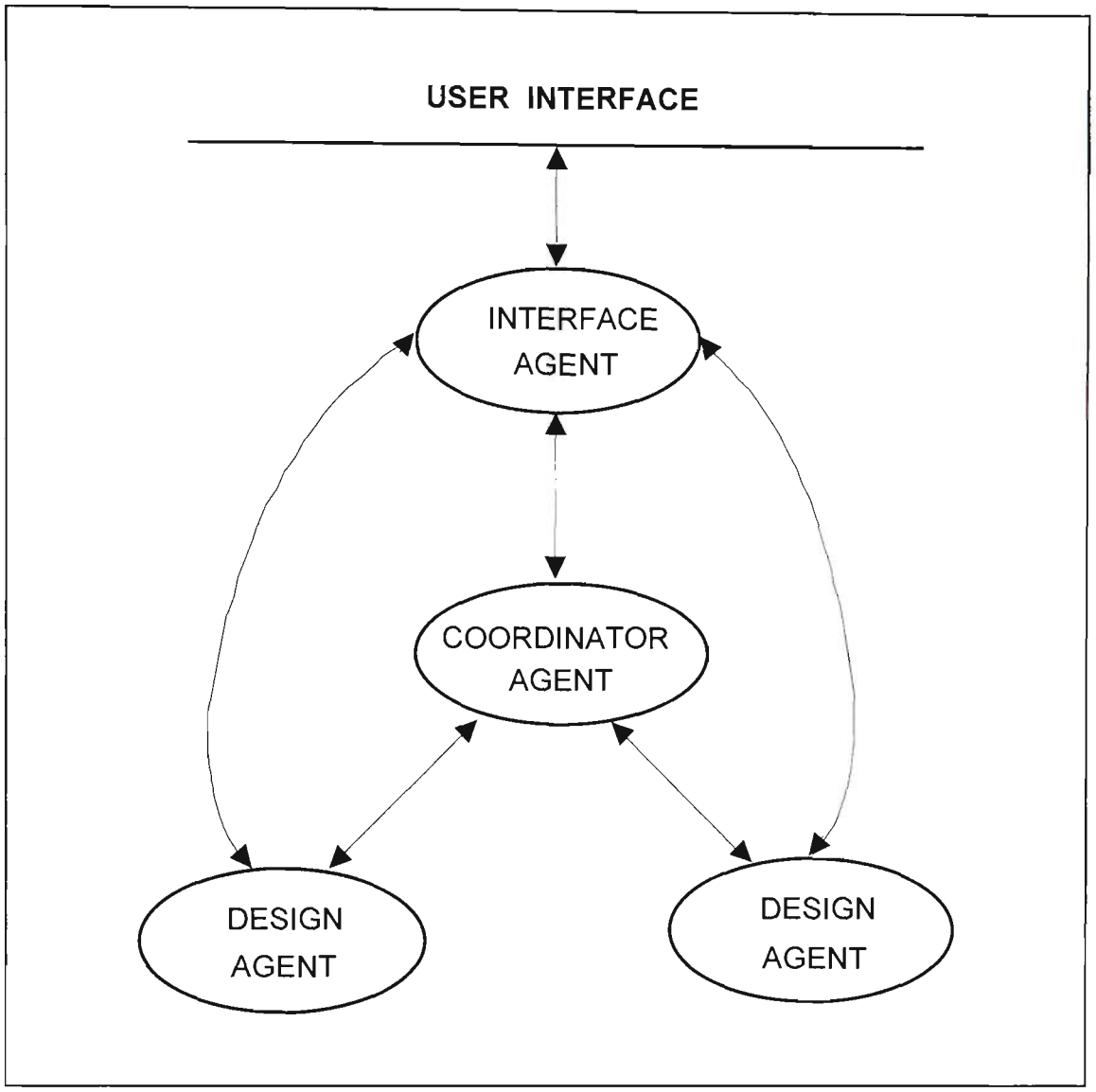


Figure 6.2 The Flow of Interactions between Agents

As illustrated in Figure 6.2, the Coordinator agent is the 'central' agent where the eventual results from the activities of the Design agents are managed by the Coordinator agent. The Coordinator agent does not control or supervise the activities of the Design agents. This is because the Design agents are provided with sufficient knowledge in which they are capable of supervising and scheduling their own activities when they are assigned a task[2].

[2]The agent architecture has been presented in chapter 3 where issues surrounding the components contained in each agent and how they function as an integrated unit has been discussed at length.

Figure 6.2 also shows how the Interface agent interacts with the user and communicates back to the Coordinator agent regarding the user's requests. In this instance, the Coordinator agent which maintains a case memory of its own, would first, make an attempt to retrieve from its memory any design case that is similar to the present problem case[3]. When found, it will be adapted if necessary, to suit the functional requirements of the current problem. Otherwise, if no similar design case is found, the problem case is decomposed into smaller tasks. A task represents a sub-problem in a particular area of the domain. These tasks are distributed appropriately among the various Design agents.

The Design agent takes on a similar problem solving approach as the Coordinator agent - it first attempts to retrieve from its case memory a case similar to the assigned task. If found, the retrieved case will be adapted. Otherwise, it tries to solve the problem by reasoning from first principle basis. Upon completing the task, the Design agent sends the solution back to the Coordinator agent.

When the Coordinator agent receives all the solutions from all the Design agents (which are presently involved in the current problem solving task), it then executes the necessary actions to integrate and coordinate all the solutions into a single and coherent solution. This solution is then posted to the Interface agent for presentation to the user.

[3]The problem case here denotes the current problem with constraints and a set of functional requirements that needs to be fulfilled and satisfied.

6.2.2 System Design

The general organisation and architecture of the system are presented in relation to the system which has been implemented in a knowledge based system. The architecture of the system has also been detailed in reference [Wong and Kalam 96, Wong *et al.* 96]. The system consists of a group of loosely coupled and decentralised problem solving agents. It has two Design agents, namely Bus agent (BA) and Line agent (LA) - each specialising in bus and line components of a power system respectively.

In short, the system consists of the following agents :-

Interface agent (IA)

Coordinator agent (CA)

Design agents (DA)

The arrows show the system components participating in the communication acts as well as the direction of the message flow. Message passing paradigm is employed as the mode of communication. As shown in the diagram, communication mainly occurs between the Coordinator and other agents. In addition to that, the Design agents can also communicate with the Interface agent when they require additional or more detailed information from the user regarding specific information or parameter values of the problem to be solved.

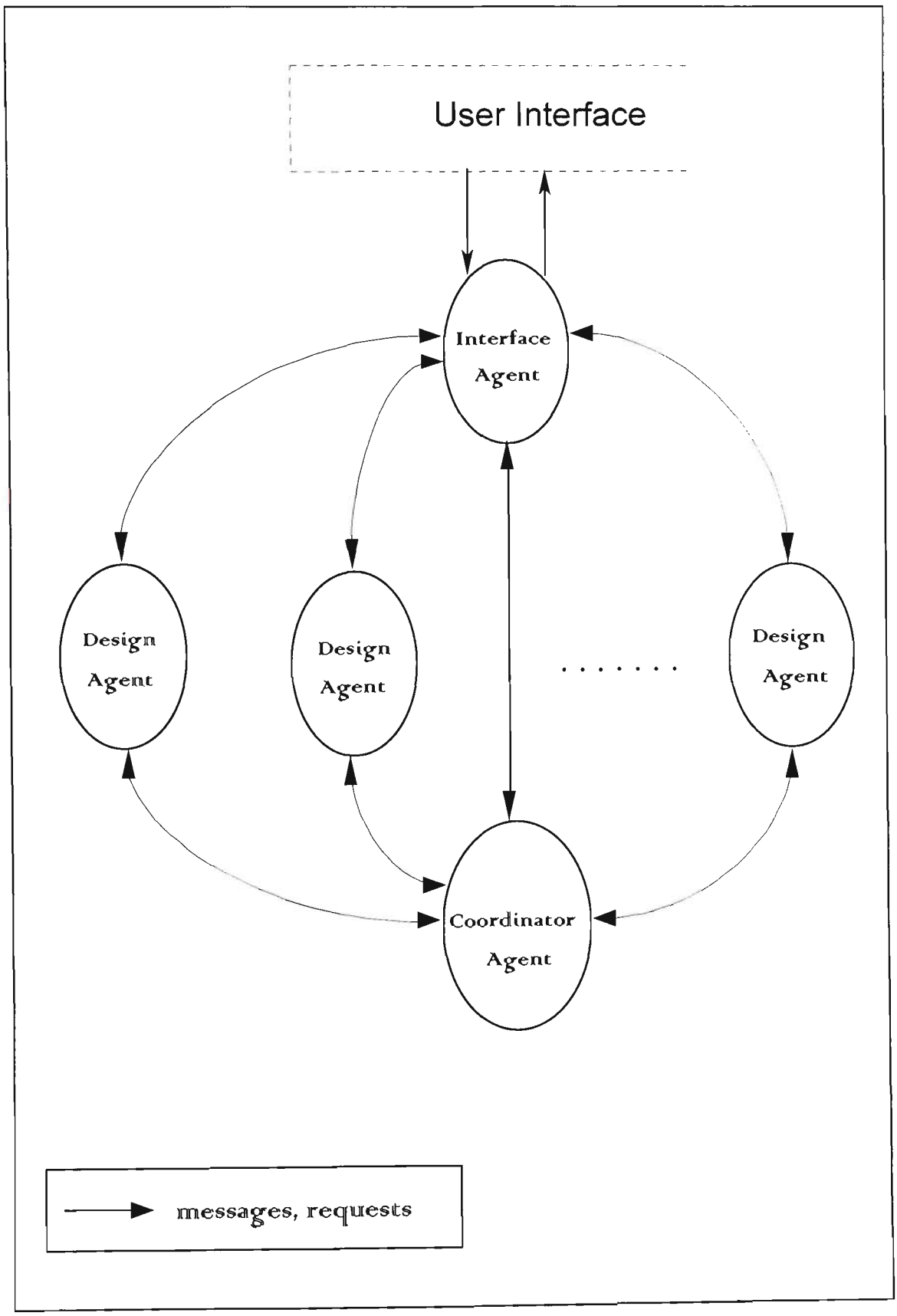


Figure 6.3 Architecture of a Distributed Knowledge Base System

In general, the responsibilities of the agents include :-

- Designing a solution case;
- Matching the current problem with similar previous designs and retrieving them from the case library. The retrieved designs may have to be adapted to suit the requirements of the current system;
- Coordinating and integrating individual partial solutions into a coherent design solution to the original problem;
- Interacting with the user intelligently. The user should have the option to review the designs, request their modification and ask for an explanation to the derived schemes.

There is a clear division of responsibilities among the agents in the system. The Coordinator agent is responsible for decomposing of the original problem and allocating the sub-tasks to the different Design agents. It is also responsible for coordinating the various design schemes into a single and integrated design solution. The duties of each Design agents are designing solution schemes in their specialised field of the problem domain As interactions with users are involved, the Interface agent will be residing at the front end of the system. Its responsibilities include communicating the designs to the user, giving them a choice to review and approve the designs. The Interface agent is also able to request detailed information about the current problem from the user on behalf of the other agents as the design progresses.

For a discussion on similar agent architectures, refer to [Genesereth and Ketchpel 94].

6.3 Design of a Distributed / Shared Knowledge Base System

In this system architecture, the domain knowledge is not only held exclusively by individual agents resulting in the knowledge being distributed among the agents, but knowledge which is common to all agents are also shared as well. The main purpose of knowledge sharing is to reduce the need for duplication and hence redundancy in the system.

The system architecture discussed in this chapter, employs a variety of different methodologies and reasoning techniques, in an attempt to maximise the benefits and advantages offered by the employment of the various different methodologies and techniques. The architecture is based on distributed expert systems, also known as agents, which operates in an object oriented environment. It employs distributed problem solving technique and multiparadigm reasoning strategy to increase its power and capacity to reason. Such employment also provides a satisfactory solution or explanation to a problem. These reasoning strategies include case based, explanation based, rule based and argumentation (reasoning under uncertainties) [Huang *et al.* 94].

6.3.1 Distributed Approach and Agent Technology

Distributed problem solving is a sub-area of AI that concerns with distributing control and data to achieve cooperation, coordination and collaboration among the agents. One of the most important constituent to the success of a multiagent system in achieving coordination is communication. Generally, communication can be achieved via message passing and/or

broadcasting. In this distributed/shared knowledge base system, communication among agents is performed using both forms of communication, where broadcasting is realised through the use of a blackboard system.

6.3.2 Object Oriented Technology

Agent based software engineering is often compared to object oriented programming [Genesereth and Ketchpel 94]. Both approaches provide a message-based interface independent of its internal data structures and algorithms but with the primary difference in the interface language. The concepts are compatible, thus the creation of software agents can be achieved more easily using object oriented programming. Moreover, most things in this world can be viewed as an object (e.g., a person, a house, a design, etc). An object or in this example, a design work can be related to a case or an object. For example, consider a designer who keeps all designs separately in different folders. Each folder contains one design with all related information and specifications. Here, a folder corresponds to a case.

Based on the above viewpoint, the object oriented paradigm would then provide a suitable platform or environment for the development of a multiagent system. In addition to the inheritance, polymorphism and encapsulation properties, reusability of software components has made the modelling and the implementation of the system much easier to achieve. All the attributes, specifications, behaviour and properties of an object can be efficiently encapsulated within the object. Even the knowledge base or the domain knowledge can also be neatly packaged within the object.

To build a good and manageable system, addressing the issue of modularisation alone is not sufficient. To make it programmable, a proper paradigm has to be selected. According to Vranes [Vranes 95]:

> *"A programming paradigm can be thought of as a basis for a class of programming languages, as an underlying computational model, as a primitive set of execution facilities, or as a powerful way of thinking about a computer system."*

The term 'object orientation' is a popular keyword and, object technology together with object oriented databases (OODB) have been the key topics especially in many current database and geographic data management research [Gunther and Lamberts 94]. The primary interest in object technology arises from its capacity to be an essential material from which large and long lived application systems, known as Persistent Application Systems (PAS) are built [Atkinson *et al.* 93]. Such systems include CAD systems, geographic information systems, urban planning systems and health care management systems. Object oriented programming paradigm assures large benefits in those delicate requirements of the software life cycle, such as reusability and maintainability of software components.

For discussions on the other reasoning paradigms employed by the system, such as case based, rule based, explanation based and argumentation, refer to *Chapter 5, Section 5.4.*

6.3.3 Blackboard Architecture

The Blackboard structure forms an integral part of the system. All agents have access to the Blackboard - they can read from and write to it. In other words, the Blackboard is a global database where partial solutions delivered by the various Design agents are recorded. It also collects and organises all partial and complete solutions generated for problem solving. Most Blackboard architecture consists of three major components [Nii 86]:

- knowledge sources;
- blackboard data structure;
- a control.

However, the Blackboard in this system is used as a medium for communication and as an integration platform to facilitate problem solving (i.e., as a platform for the Coordinator agent to coordinate and integrate partial solutions into a single coherent design solution). Therefore, to sufficiently accommodate the requirements of the system, the Blackboard here consists of two components only - *Message Area* and *Data Store*.

The *Message Area* is the area that records all current data and information with regards to the problem case such as the identities of the agents involved, the current status of the problem solving situation and whether coordination is required. The *Data Store* are organised in layers. Each layer contains the current partial solutions posted by a particular Design agent, which would be eventually coordinated by the Coordinator agent into a coordinated and integrated complete solution.

6.3.4 Agent Architecture

As mentioned earlier, the architecture and organisation of the system here is different from the system architecture described in the previous section. However, the responsibilities of each agents here are not too different from the agents in the previous system. They both have similar type of agent architecture. In the earlier system (a distributed knowledge base system), all interactions between the agents and the user are achieved via message passing to the Initiator agent. However, in this current system, all agents including the Design and Coordinator agents are allowed to interact and communicate with the user directly.

Object oriented processing provides support for data abstraction, knowledge encapsulation, inheritance, reusability, extensibility and modularity. Hence, the message passing capability of object oriented processing allows the system to keep knowledge about the domain separated from the knowledge about reasoning. Using this model, agents can be structured using classes.

Figure 6.4 shows the architecture of the Initiator and Coordinator agents in the system.

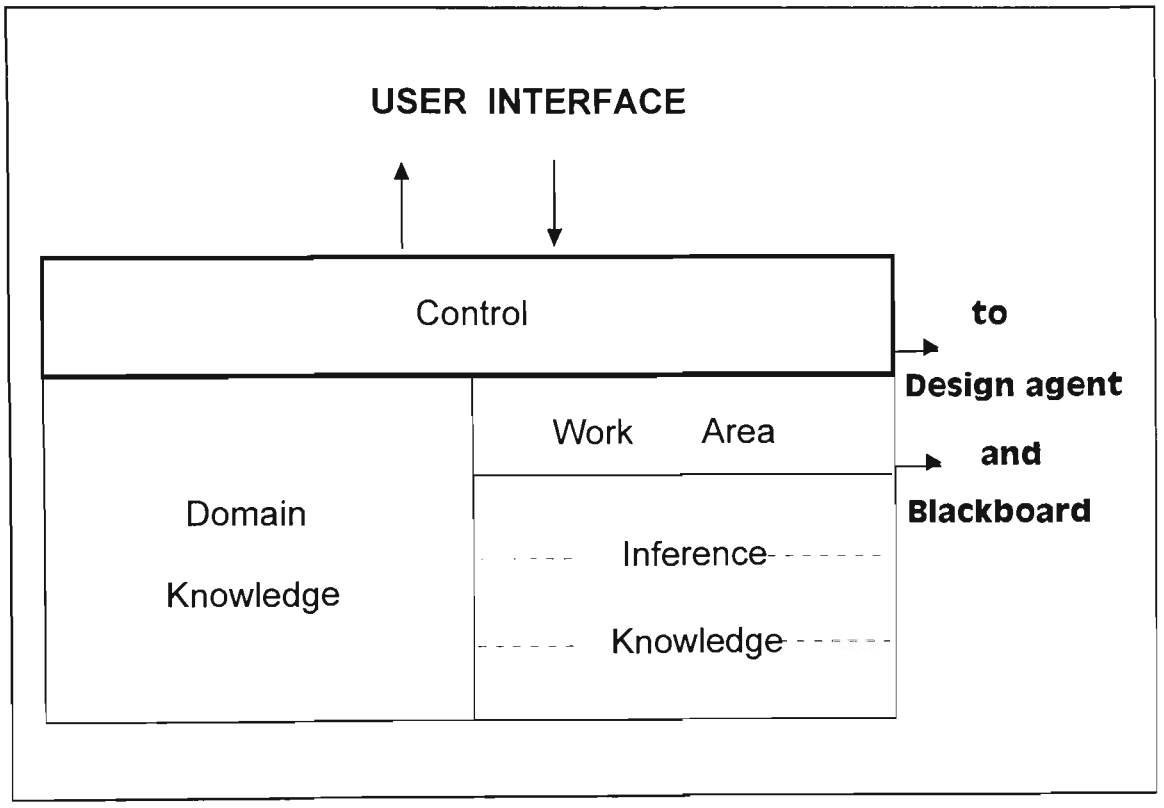


Figure 6.4 Architecture of Initiator and Coordinator agents

Figure 6.4 shows the main components that made up the agent. The diagram illustrates the agent architecture as consisting of separate components or modules. However, these components are actually implemented as layers. The reason for the different diagrammatic representation is because conceptually, it is easier to understand how each layer/component functions and interacts with one another. Each component serves a specific purpose, either to carry out certain responsibilities or to act as a repository of knowledge and information.

The architecture of the Design agent is slightly different from both the Initiator and Coordinator agents. The architecture of the Design agent is shown in Figure 6.5.

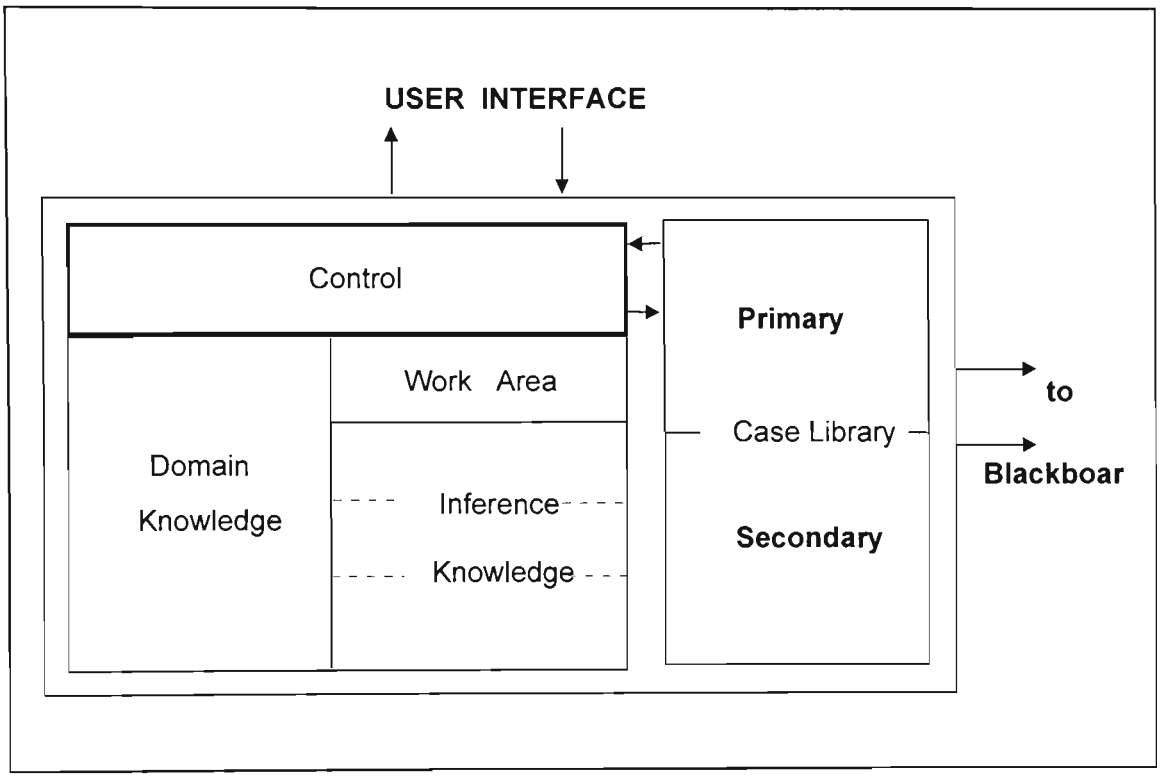


Figure 6.5 Architecture of Design agent

The Design agent has an additional feature - a memory area which is regarded as the *Case Library* - that stores historical cases. This memory represents the agent's acquired and accumulated experiences.

The *Case Library* is divided into two memory sections: *primary* and *secondary*. The *primary* section stores the verified and proven cases in practice, whereas the *secondary* stores cases that have not been implemented, tested or verified before. As stated earlier, the reason for this separation is to ensure the reliability of the solution is maintained. Solutions retrieved from the *primary* section are identified as reliable and safe because these solution cases have a history of successful application. Therefore, retrieval is always carried out from

the *primary* section. However, in situations where there is no similar cases found, retrieval is then proceeded to the *secondary* section[1].

Another aim of the organisation and memory management of the *Case Library* is to provide a more efficient and faster retrieval process as the search space is reduced by the division of the memory. Furthermore, the cases retrieved can be categorised by its reliability which is an important and determinant factor before the actual implementation of the design is carried out.

6.3.4.1 The Components of the Agent Architecture

The agent comprises the similar components as the generic agent architecture introduced in chapter five. In addition to a *Domain Knowledge*, an *Inference Knowledge*, a *Control*, a *Case Library*, it also has a *Work Area*.

The *Work Area* is a place where temporary data are registered during problem solving. All data and information passed from the external source to the agent are stored and kept in the *Work Area* during problem solving. All components have access to the *Work Area* to enable each of them to use the data and information required to perform their respective tasks.

The *Control* controls and schedules the agent activities. It is also responsible for calling the execution of the appropriate reasoning strategy (kept in the

[1] The cases retrieved from the secondary memory should carry a warning note that the design has not been implemented or tested before. It is then up to the user to decide on the reliability of the presented design case.

Inference Knowledge) and applies the *Domain Knowledge* to solve the current problem.

The *Domain Knowledge* represents the agent's knowledge and expertise in a specific area of the application domain. The way this knowledge component is represented depends on the *Inference Knowledge*, which is the knowledge applied to task management and decision making. As shown in the above diagram, the *Inference Knowledge* is represented in layers - each layer contains one inference mechanism or reasoning paradigm employed by the agent.

6.3.5 Agents' Functions

The Initiator agent initiates the problem solving task. The agent can either communicate with the user who may specify the type of problem to be solved or alternatively, an external program could provide the agent with the information or parameters required for problem solving. The agent keeps the profiles of all the Design agents in the system and communicates with them through message passing. The Initiator agent initiates the problem solving task by decomposing the problem and assigning the various tasks to the appropriate Design agents. At the same time, the identity of the Design agents involved in the current problem solving are posted onto the Blackboard. If the information is received from an external system, the data will be filtered by the Initiator agent before the correct information are transmitted to the appropriate Design agents. If the source of data comes from the user, the Initiator agent performs the following tasks: identifies the problem, decomposes the problem into sub-problems, assigns

tasks to the various problem solvers and directs the user to interact with the appropriate Design agent.

The Design agent may represent an expert system in a part or a subset of the problem domain. Each Design agent works independently; it employs case based and rule based reasonings' techniques. It attempts to solve a problem by first applying its previous experiences. This is accomplished by retrieving similar designs from the *Case Library*. The *Case Library* forms part of the distributed knowledge which stores historical cases. However, the Design agent could also be instructed by communication with the user to bypass the retrieval process and starts the problem solving task from first principles. When no similar design cases are retrieved, problem solving begins from scratch. Either way, a case would be generated which defines the problem and contains the solution for it.

The Coordinator agent serves as the system controller and coordinator. Coordination in this context means the process of coordinating and integrating the various design cases which have been posted to the Blackboard, to form a single coherent solution. The Coordinator agent employs multiparadigm reasoning strategies consisting of explanations, rules and argumentation to perform its task. The coordination process may involve refinements or modifications and adaptations to some of the designs. However, the modification process will fail when one or a few of the design cases are too costly or complicated to be modified. Similarly, the modifications will not be carried out if they have an adverse effect on the other design cases. In either situations, the Coordinator agent would send the case(s) back to the relative originating Design agent(s), informing them the cause of failure and requesting

for the case(s) to be rebuilt. Subsequently, the respective Design agent would carry out its responsibilities and repeat its design process again. A new case is then constructed and posted to the Blackboard once more. Once an integrated solution is built and presented to the user, he/she may request for an explanation as to how the particular scheme was derived. The system architecture is shown in Figure 6.6.

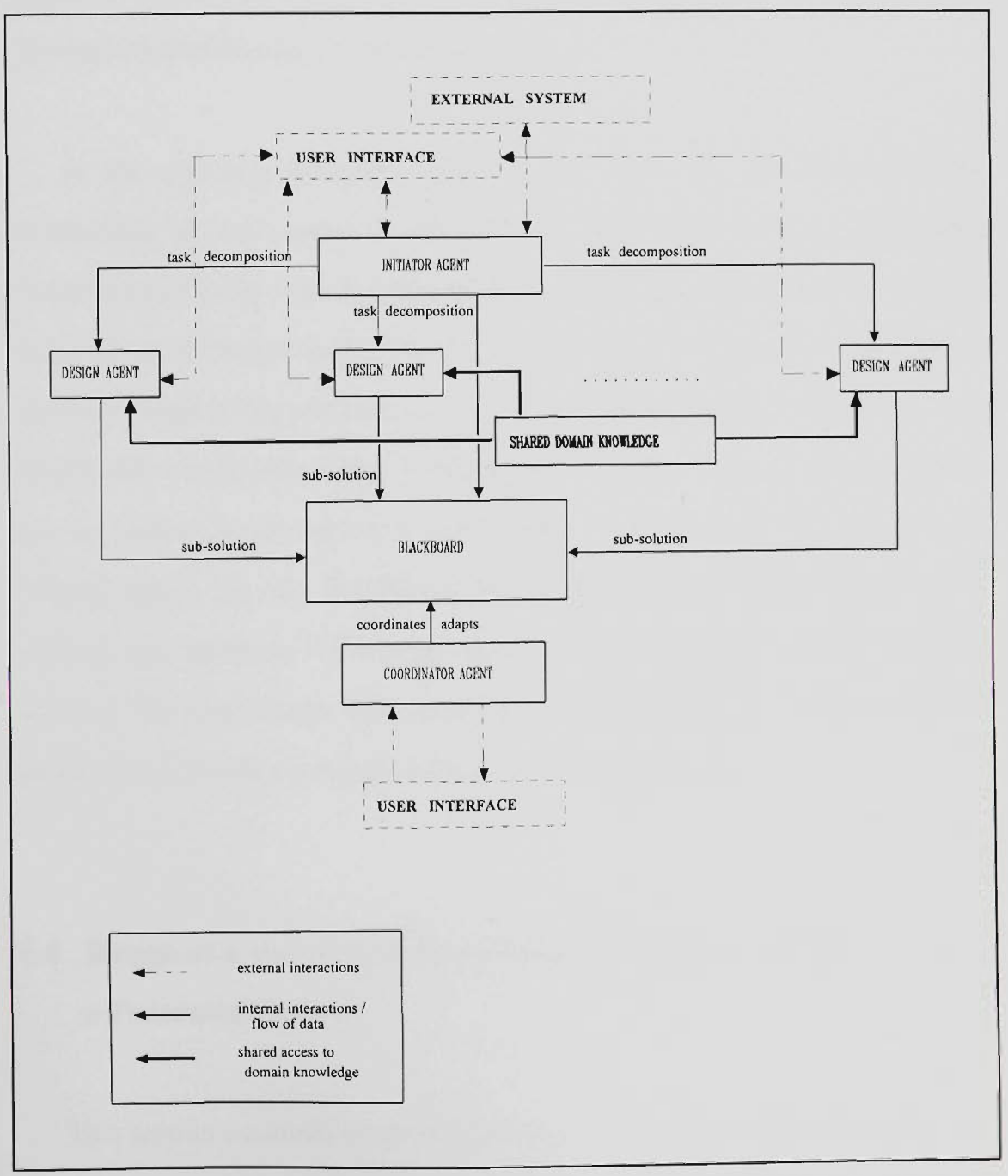


Figure 6.6 Architecture of a Distributed/Shared Knowledge Base System

In a hierarchical organisation, the overall system control is given to a central controlling agent, which in this system, is given to the Coordinator agent. The rest of the agents have minimum control - which is restricted to controlling its own activities. On the contrary, agents in a heterarchical organisation, are mainly known as the autonomous agents, which have equal amount of control in the general problem solving activities. In the system of autonomous agents, a center of controlling entity could be dispensed of. Hence, there is no need for a specialised coordinating or controlling agent.

In the system architecture illustrated in Figure 6.6, the shared Domain Knowledge in the system is accessible by all Design agents. The Shared Domain Knowledge contains knowledge about the application domain which is common to all Design agents. The Design agent may need access to this part of the knowledge during problem solving or when answering to user's query. The Blackboard on the other hand, which is used as another form of communication can be utilised by all agents in the system. While the Initiator agent and the Design agents use the Blackboard for information recording purposes (i.e., writing and updating information about the status of the current problem solving), the Coordinator agent employs it as a platform for coordinating the partial solutions into an integrated and coordinated solution.

6.4 Design of a Distributed Knowledge Base System within a Federated System

This section examines on another alternative architecture which employs the generic agent architecture within a federated system to develop an intelligent

multiagent system. In a federation architecture, the agents surrender their autonomy to their local facilitators. The knowledge bases and information are distributed among the collaborating agents.

In a federated system, the agents are organised into several communities. The agents form a 'federation' in which their communication is assisted and coordinated through system programs called facilitators [Khedro *et al.* 93]. It is the responsibility of the facilitators to determine the appropriate recipients of the messages and forward the messages accordingly. They also handle a number of important operations to facilitate communication and exchange design information and knowledge among agents. In performing their responsibilities, they also translate message, schedule the executions of different agents, help to decompose the problems into sub-problems, and assist in relating different design information [Khedro *et al.* 93]. An illustration of a typical federation architecture is shown in Figure 6.7.

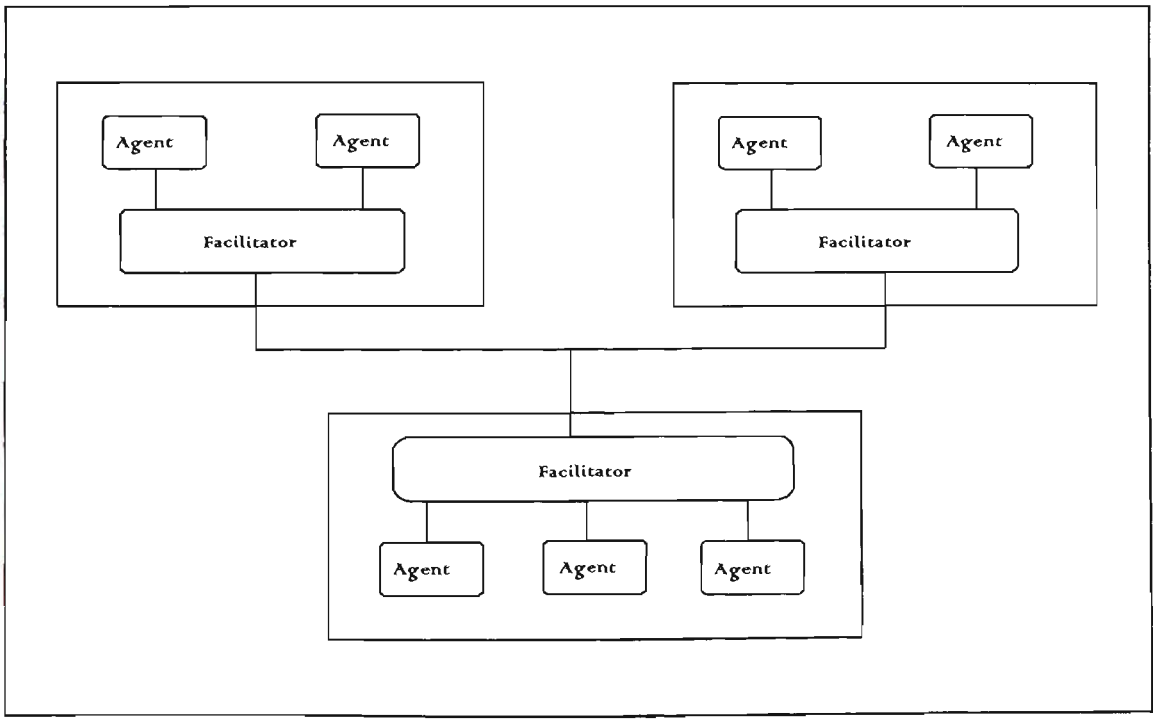


Figure 6.7 Example of a Federation Architecture [Khedro *et al.* 93]

The architecture presented in this section is an agent based system implemented in an object oriented environment. It operates in a federated system where the agents coexist with other entities in the system. These entities include a Initiator, Facilitators, Agent-Generators, a Controller and a Modifier.

6.4.1 The System's Agents and Modules

The system is basically built up of libraries containing past design cases. Each case contains the knowledge and information of the designs in the problem domain. The cases, represented as objects using object-oriented concepts, are the agents in the system. They have a state and behaviour and, encapsulate not only data and information but knowledge as well. The design of the system can also be referenced in [Wong *et al.* 95].

The operations of the system are controlled by several entities/modules which include the Initiator, Facilitators, Agent-Generators, Modifier and Controller. In addition to that, the system has several components. Each component can be viewed as a community that houses a specific group of agents. However, there are no intra- or inter-communication among these agents, that is agents within a community do not communicate with one another nor with agents from other communities. Communication is performed via the local Facilitators as in federation architecture [Khedro *et al.* 93] and coordination is achieved by the system's Controller.

The following subsections present a more complete description of the agents and entities of the system.

6.4.1.1 Agent Architecture

The agents are embedded in an object-oriented environment and communicate using an object oriented language. Each agent can be viewed as a knowledge based system with an inference engine and a knowledge base. In other words, an agent is not only a sophisticated data structure but also an expert system.

The agent architecture shown in Figure 6.8 depicts the five main components which embodies an agent in this system architecture. This agent architecture is very similar to the generic agent architecture. The components are allowed to communicate with one another and this is achieved via message passing. Similar to the previous agent architecture, the components here are implemented in layers. However, they are regarded as components diagrammatically because they are easier to understand conceptually.

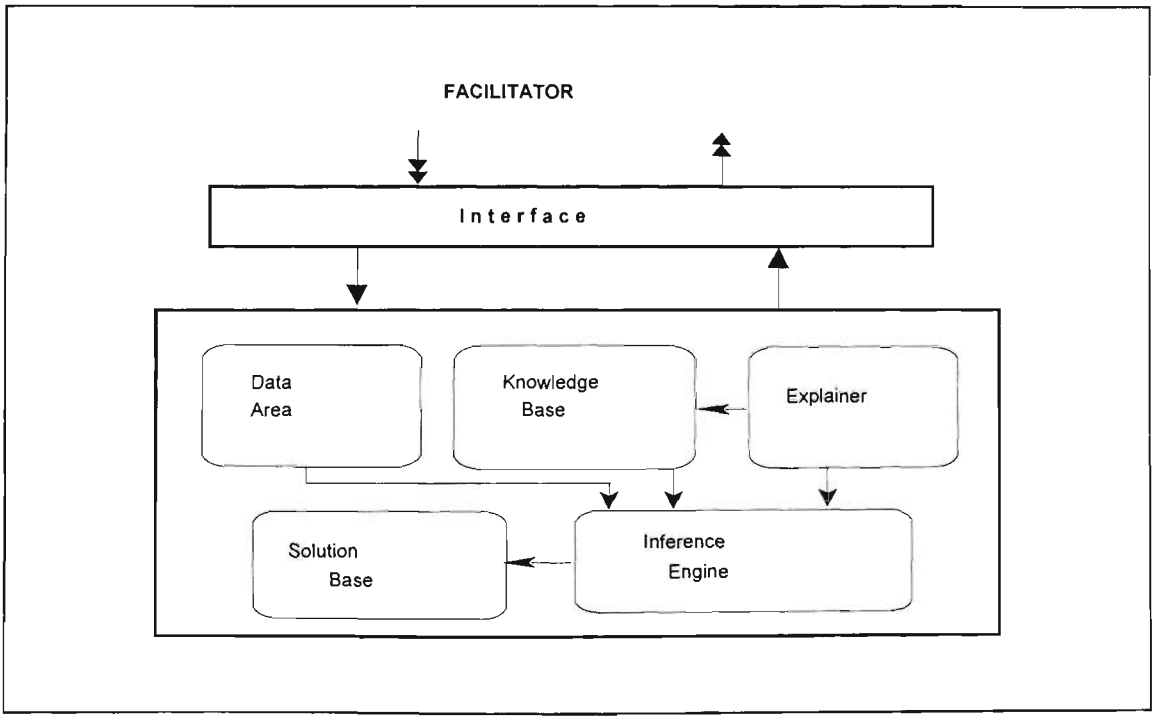


Figure 6.8 Agent Architecture

The purpose of the *Interface* layer is to smooth out the communication between the agent and the Facilitator. The *Interface* layer is not required to perform interpretation or translation of messages passing to or from the agent. However, if communication is achieved in a common language within the same platform, this layer becomes optional. In addition, the *Interface* layer is expected to communicate and relay any messages transmitted back and forth between the agent and the local Facilitator. The data occupying the *Data Area* component at any one time are only temporary. These data represents the current information and design specifications which aids in the problem solving task.

Domain knowledge are facts about a particular area of the application domain which makes the agent an expert in the specialised domain. This knowledge is maintained in the *Knowledge Base* where it is being used and applied by the *Inference Engine* to construct a solution. These solutions are accumulated and saved in the *Solution Base*. In other words, the *Solution Base* stores design solution to a specific problem. This component can be related to the Case Library component (memory area) in the previous agent architecture. Except for the terminology which is used here, the *Solution Base* does not differs much from the Case Library. Both components store previous solution cases experienced by the relative agent. However, in the *Solution Base*, there is no distinction between successful or failed cases. This base contains a number of pointers or references to a database where detailed information is stored. This pointer feature provides the agent with a rich source of up-todate information, while eliminating data duplication and encouraging a more efficient memory utilisation at the same time.

The *Explainer* component contains a number if inference mechanisms which could manipulate the knowledge sitting in the *Knowledge Base* area in order to formulate an appropriate answer to a query. In short, the *Explainer* provides the explanation which justifies for the proposed design scheme.

6.4.1.2 Facilitators

In this system, one Facilitator is assigned to one community and they are invoked by the Initiator module. The concept of the Facilitators in the system is similar to that in a federated architecture [Devapriya *et al.* 92], however their functions are quite different.

Here, each Facilitator functions as a supervisory agent. The Facilitators do not communicate with one another directly, but rather through the Global Society. One of the main responsibility of a Facilitator is scheduling the Agent-Generator's activities and acts as a communication buffer as well. It also communicates the Initiator's request to its community. The request is a message, listing the user's specifications and requirements for a problem to be solved. Based on this message, the Facilitator selects the agents, if any, from the community that satisfy the required specifications. If none is found, it invokes the local Agent-Generator to generate a new agent which satisfies the design constraints and requirements. Once the agent is selected or newly created, the Facilitator then transports it to the Global Society for further coordination by the Controller.

6.4.1.3 Agent-Generators

One Agent-Generator is assigned to one community and its activities are controlled by the local Facilitator. The main responsibility of the Agent-Generators is simple and direct. Given the functional requirements and specifications, it generates and builds new agents that meet the constraints of the problem that is being solved. The task of generating new agents becomes necessary when the existing agents of the particular community is unable to fulfil the requirements and specifications of the current problem. The new agent is added to the community automatically. As such, it can be seen that the aforementioned process has a positive effect on the population of a community.

6.4.1.4 Modifier

There is only one Modifier in the entire system and it resides in the Global Society. The module is only invoked when the selected agents in the Global Society can collaborate but cannot coordinate with one another. Therefore, some modification would be required to alter or modify the values of the specification. Such process can give rise to a new agent if the modified agent becomes too different from its original form/specification. In other words, the Modifier may also have a positive influence on the population growth in a community.

6.4.1.5 Controller

There is only one Controller which acts as a manager in the entire system. The Controller is responsible for the organisation of and communication among the agents. It also controls the activities of the Modifier. The primary task of the Controller is to coordinate the different agents posted to the Global Society, in order to construct an integrated complete solution to the user's problem. The solution in fact, may represents the design solution based on the constraints given by the user. This coordinated design is saved in the Global Community. However, when one or more of the designs could not be coordinated, the Controller would invoke the Modifier to modify the designs concerned.

However, if the user is only interested in a partial solution which can be provided by one agent, then coordination is not required at all. Therefore, the Controller will not be required to perform its task. In such circumstances, the function of the local Facilitator alone will be able to produce the output required by the user.

6.4.2 System Design

The federated multiagent architecture introduced here is shown in Figure 6.9.

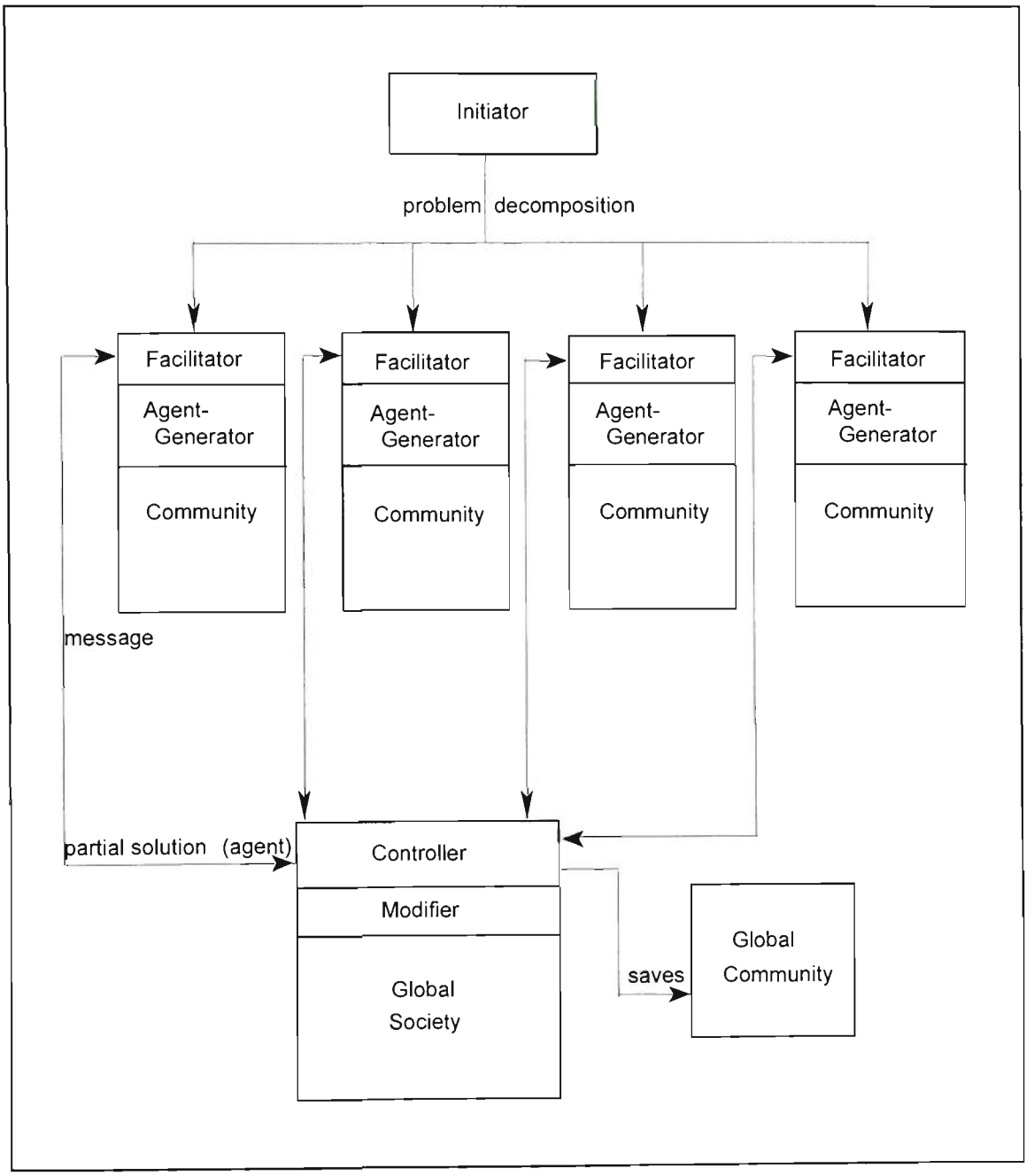


Figure 6.9 Architecture of a Distributed Knowledge Base System within a Federated System

The design system architecture is based on the application requirements and the system descriptions. The multiple and distributed problem solving agents described in this architecture exist within the one homogeneous system. While communication is achieved through the local Facilitators, the agents' actions are coordinated by the Controller to achieve a common objective.

The agents are categorised and grouped into different communities. Each agent belongs to one community only. The creation of an agent is dynamic and an agent is only created or generated if there does not already exist a similar agent in the community. The communities in the system are the database areas which are represented locally and globally. Agents of the same type are found in a local community whereas in a Global Community, groups of different agents can coexist.

6.4.2.1 Organisation of the System

The system is organised hierarchically with the Controller being the main controlling entity in the system. The Initiator is the interface entity which initiates problem solving by interacting and communicating with the user or an external system. Similar agents are assembled in one community which is supervised by a local Facilitator. In addition to that, the local Facilitator also controls and supervises the activities of the Agent-Generator.

Problem solving begins with the user interacting with the system through the Initiator regarding the problem to be solved. The Initiator then decomposes the problem and sends requests to the appropriate Facilitators to solve the decomposed problem. The Facilitator then sends the selected agent, if retrieval is successful, or otherwise, a new agent to the Global Society. The Global Society is actually managed by the Controller. It can be regarded as an assembly base or area for all selected or new agents which would eventually form the final solution.

The messages communicated by the agents are the design information based on the constraints and specifications provided by the user. The design information represents the partial solution to a problem. It is the responsibility of the Facilitators to convey the partial solutions represented by their respective community to the system's Controller. The responsibilities of the Controller include coordinating, correlating and eventually presenting the coordinated design information to the user. This is one of the main advantages of having Facilitators and a Controller in the system. It improves the flexibility in the integration of agents, which allows the agents to be ignorant or indifferent to other existing agents in their own community as well as in the other communities in the architecture. This eliminates the task of informing the agent or updating its knowledge.

Each Facilitator has to transport at least one agent as a representative of its community to the Global Society. The agents in the same community compete with one another, while agents from differing communities complement one another. Therefore, if the existing agents' specifications in one community do not match the essential requirements, a new agent has to be created. In such circumstances, the Facilitator sends a message to the Agent-Generator. It is then the responsibility of the Agent-Generator to generate the new agent which fulfils the required specifications.

The Controller regulates the interactions among the selected agents which are posted to the Global Society, based on a collection of constraints and constructs the solution. The solution is the result of a coordination of partial solutions represented by the various agents from different communities. Given a

set of constraints and specifications, it is possible to generate more than one solution to a problem.

6.5 Conclusion

This chapter has described three different system architecture designs by organising the agents in the systems of different environment. The main aim of this chapter is to study and investigate how the system architectures can change with the change in the autonomy of the agents. It can be seen from the above presentations that by applying the adaptive generic intelligent agent that we have created, we could reuse the agent and/or modify it to build different types of agents. These agents can then be employed, reused and reorganised in various ways to develop different system designs to suit a variety of different environment, system requirements and constraints.

The system architectures and methodologies presented here are applicable to a wide range of applications, particularly in decision support systems area. A prototype for each of the system designed has been successfully developed, which meets the design requirements. The implementation of each system is discussed in the following chapter.

Chapter Seven

SYSTEM IMPLEMENTATION

ABSTRACT

This chapter presents the implementations of three system architectures for the development of an Intelligent System for Power Protection (ISPP). ISPP is a multiagent based system which generates protection schemes for a power system or sections of a power system. The ISPP is intended to assist engineers in the design, selection and analysis of protection schemes. At the same time, it could also be used as a training or learning tool for new graduates and inexperienced engineers in the protection area. This chapter focuses on the implementation of the systems which have been designed in the previous chapter. A prototype for each system has been built, and demonstrated to work successfully in the design of protection schemes.

7.0 Introduction

In this research, the multiagent methodology is applied to power system protection as it represents a class of interesting and varied design problems. There are very few research projects undertaken involving applications of agent technology to power systems. Examples of such applications can be found in the distribution, transmission and supply of electricity [Cockburn and Jennings 94]. However, as far as literature study reveals, there has been no similar research projects done in the area of power system protection.

As discussed in chapter two, the design of a protection scheme for a power system or a part of power system is not an easy task. The protection scheme must be designed effectively so that the scheme meets the reliability

requirements, speed and selectivity as set by the system operating constraints [Wong *et al.* 94]. The process of power protection design requires knowledge of previous designs (i.e., past experiences), use of 'rules of thumb' and heuristics plus a comprehensive knowledge of the protection and related areas. To solve the design problem, the engineer will most likely need to switch from one paradigm to another in the course of the solution. In other words, the protection engineer must not only be equipped with the knowledge regarding the protection area but he/she must be able to employ various reasoning paradigms and be able to select the most appropriate paradigm for the task.

To assist the protection engineer, a multiagent system is introduced. The system consists of a collection of interacting agents which cooperate to solve design problems by communicating with one another. These agents include interface agent, coordinating agent and design agents. A design agent is an expert in a particular component of a power system. A component refers to a busbar, line, generator or other part of a power system which requires protection. Each agent specialises in designing protection scheme belonging to its area of expertise. The coordinating agent is responsible for coordinating various design schemes produced by the design agents into an integrated and coordinated protection scheme for a power system. The interface agent is introduced to mainly provide an intelligent interface between the user and the system.

7.1 Intelligent System for Power Protection (ISPP) - System Requirements

The system is designed to aid the protection engineers in the design of protection schemes for a power system. A protection scheme could be defined as a collection of necessary and coordinated protective gears[1] which are set to operate within a time frame when a fault occurs in a power system. The system is not a real-time system, however, it is required to function effectively and reliably to produce accurate result. In other words, time is a secondary factor compared to the correctness of the output. It is vital and critical that the scheme is designed appropriately to provide sufficient protection to all parts of the power system because an incorrect or even a delayed operation of the protection system can be disastrous. That is why, in practice, engineers are hesitant to recommend or apply any protection scheme that have not been proven or verified before.

There are a number of environments that could be used to develop a software protection system. The environment, as well as the system requirements and constraints, influence the development of a system. In the implementation of the three systems, two types of environment are used: knowledge based and object oriented.

As the previous chapter has investigated on how the change in the agent's autonomy could produce different system architectures, this chapter continues with the focus on the application and implementation of these systems to power system protection. The following sections illustrate how the generic agent

[1] Protective gear could include circuit breakers, fuses and relays.

architecture that we have designed and modelled in this thesis can be applied adaptively to develop systems which are designed for different environments. These systems also have the advantage of being able to be developed incrementally.

7.2 Applying Agent Technology in Developing an Intelligent System for Power Protection (ISPP)

The systems architectures developed for ISPP represent the different approaches in the integration of diverse knowledge representation and multireasoning techniques for the purpose of solving and designing protection schemes for a power system.

The integration of different techniques and paradigms yields an environment that is more effective, flexible and robust than the use of any one single technique in isolation. In other words, the user of multiparadigm approach is much more effective in building a robust system than just relying on just one paradigm/technique. This can be seen from the conventional rule based systems which have proven to have many limitations. The primary mode of communication used by the agents is the message passing paradigm. In later sections of this chapter, a system architecture which applies and incorporates the intelligent agents using alternative mode of communication, which includes the use of blackboard as a global base for composing and coordinating solutions is also presented.

7.3 Distributed Knowledge Base System in a Knowledge Base Environment

7.3.1 Environment and Software

The environment in which the distributed knowledge base system was developed consists of Lucid Common Lisp 4.1 running on SPARC Sun OS 4.1.x under Open Windows 3.0. In addition, the following two libraries of Common Lisp subroutines have to be loaded into the working Lisp environment: Epilog and API.

7.3.2 System Flow of Control

The control flow of the system can be illustrated using a flow diagram as shown in Figure 7.1.

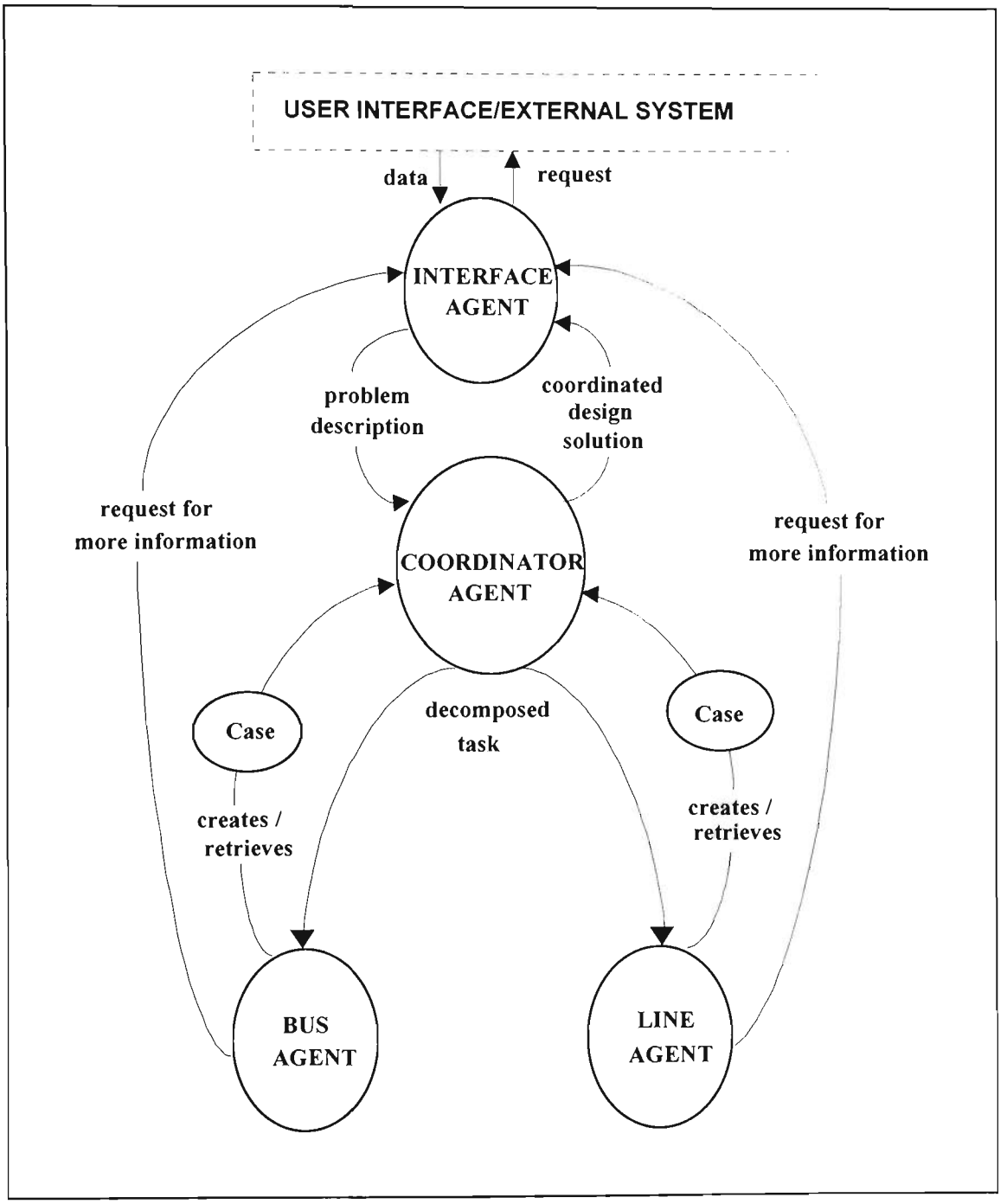


Figure 7.1 Control Flow of the System

The system interacts with the user or any external system via the Interface agent. The Interface agent passes the user's problem specification to the Coordinator agent, which then decomposes the problem into smaller and more manageable tasks. For example, if the problem involves designing protection schemes for a bus and a line components of a power system, it could then be

decomposed into two smaller individual component tasks which would be designed separately before they are coordinated and integrated into a single solution. These smaller decomposed tasks would then be assigned and distributed to the Bus and Line agents respectively.

Each Design agent (i.e., Bus and Line agents) applies its specialised knowledge to derive the design solution or case and conveys it back to the Coordinator agent. The eventual design solution may consist of either an adapted and modified case or a new case altogether. Subsequently, when the Coordinator agent receives all partial solutions from the Design agents, it would attempt to coordinate them into one single coherent and integrated solution. This final solution is then passed on to the Interface agent which then communicates it back to the user.

7.3.3 Constructing the Agent

The agent architecture used to build the agents of the ISPP system is based on the generic architecture presented in chapter five. Using the knowledge base environment, the agent architecture is implemented in a number of layers which consist of `Communication Layer`, `Control`, `Inference Knowledge` and `Domain Knowledge`.

Each agent is endowed with the knowledge and expertise required to fulfil its responsibility so that it can perform its various functions, including task decomposition. The `Inference Knowledge` together with the `Domain Knowledge` form the agent's knowledge base.

The `Inference Knowledge` is further organised into several sublayers which consists of :-

- performative layer;
- function and procedural layer;
- primitive layer.

A selector is located within these layers - which selects and invokes the appropriate inference mechanism during the problem solving stage. The layered `Inference Knowledge` is shown in Figure 7.2.

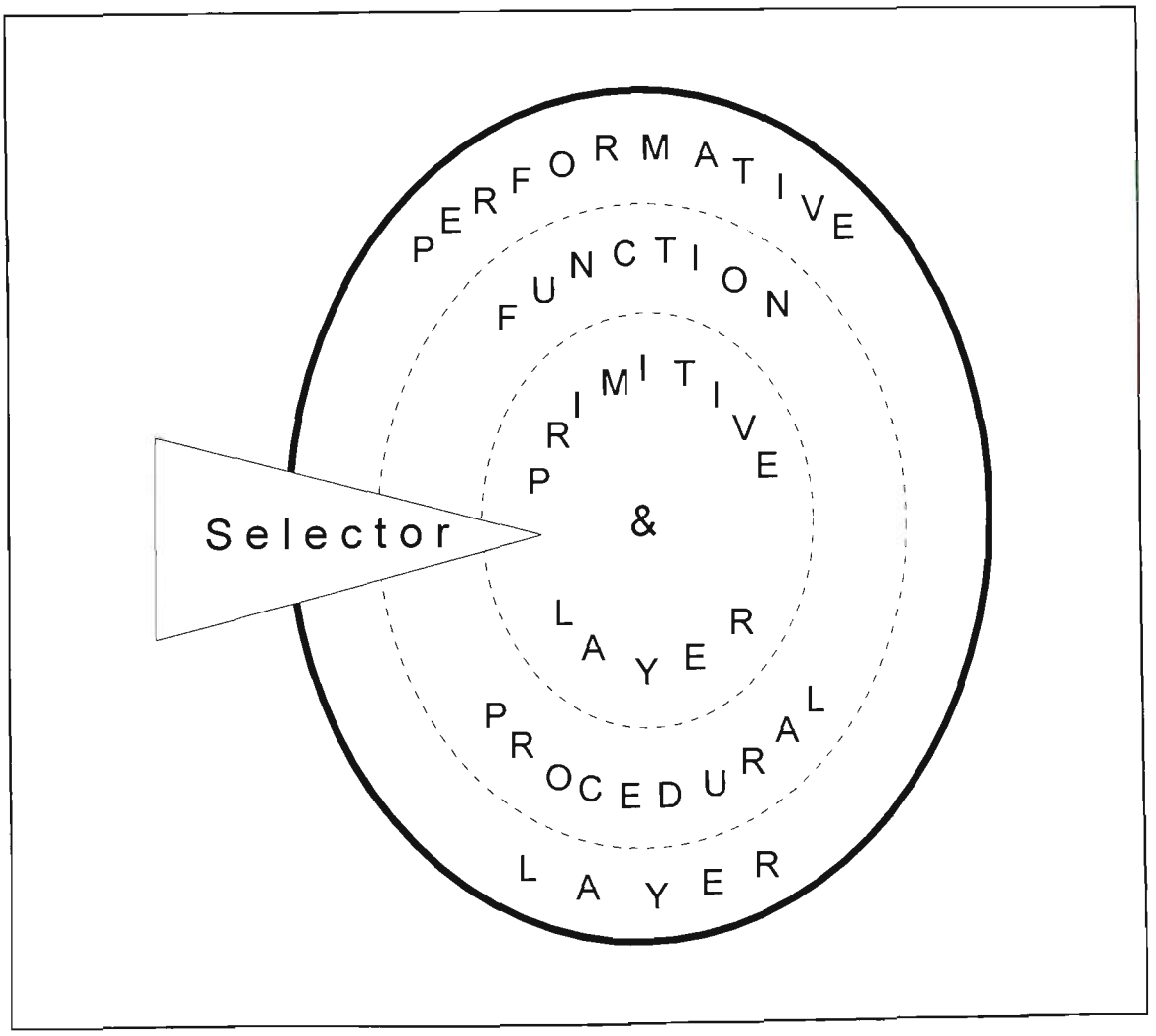


Figure 7.2 Agent's Inference Knowledge

The Domain Knowledge contains facts about the domain the agent operates and specialises in, and also contains knowledge of previous designs. The latter knowledge is kept as cases in two separate case libraries which represent the primary and secondary parts of the agent's memory. These case libraries are implemented in separate layers.

Figure 7.3 indicates how the agent's Knowledge Base, consisting of both the Domain and Inference Knowledge, fits into the agent architecture.

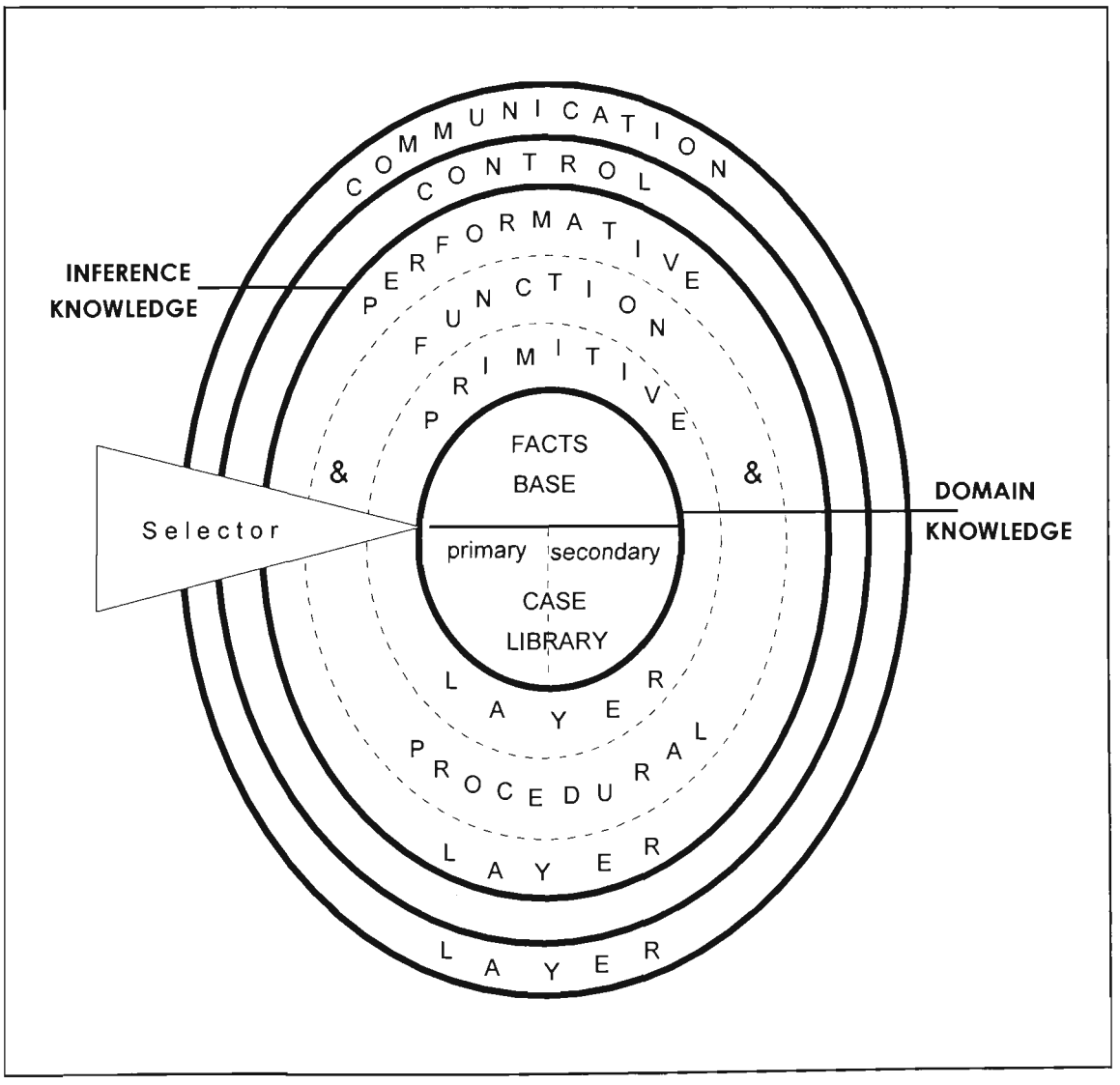


Figure 7.3 Agent's Knowledge Base

The sublayers define the various abilities possessed and which may be required to solve any problem that the agent encounters. The performative layer defines the agent capabilities in terms of task definitions. Various reasonings such as heuristics, analogical reasoning and case based reasoning are incorporated and implemented in both the function and procedural layer while the primitive layer contains search algorithms.

The agent domain knowledge, including previous designs kept in two separate case memories, is stored in separate layers in the knowledge base. The Lucid Lisp environment in which the agent is constructed, uses two additional libraries of Common Lisp subroutines[2]: Epilog and Application Program Interface (API). Epilog [Epilog 94] is a knowledge representation and inference system based on Prolog. It supports the language known as Simplified Interchange Format (SIF), which is a subset of Knowledge Interchange Format (KIF)[3]. The API [Singh 93] provides the interface between local and external agents. API also offers services such as identification of local agents, communication with API via TCP, email and Lisp, definition of performatives for local agents and maintaining connections with external agents.

Tasks are described and decomposed using the SIF. The Epilog system is used to construct and manage the knowledge bases which are represented in the form of lists containing sentences that are encoded in SIF. Epilog's powerful in-built inference procedure based on model elimination process, is used to retrieve design cases from the case library.

---

[2]For further information, refer to Appendix A for API and Appendix B for Epilog.
[3]KIF are used to refer to terms or sentences which form the agent communication language.

An example of sentences (facts) stored in the Bus agent knowledge base are:

```
(high-impedance (listof dedicated saturate common))
(low-impedance (listof notdedicated notsaturate common))
```

The above facts would be interpreted respectively as follows :-

High-impedance (differential) protection scheme is applicable provided the bus component has dedicated current transformers (ie. cts) which are of common turns ratio and which may saturate.

Low-impedance (differential) protection scheme is applicable provided the bus component does not have dedicated cts, the cts turns ratio are not equal and they do not saturate.

An extract of the Bus and Line agent knowledge bases [GEC Alsthom 87] are as follows:

```
&& BUS Agent Knowledge Base &&

;;;; Unit Protection where backup is not possible/necessary.
(save '(high-impedance (listof dedicated saturate common)) 'BusDK)
(save '(high-impedance (listof dedicated notsaturate common)) 'BusDK)
(save '(medium-impedance (listof dedicated saturate notcommon)) 'BusDK)
(save '(medium-impedance (listof notdedicated saturate common)) 'BusDK)
 (save '(low-impedance (listof notdedicated notsaturate notcommon)) 'BusDK)
(save '(low-impedance (listof notdedicated notsaturate common)) 'BusDK)
```

;;;; ***Backup Protection For DownStream.***

(save '(idmt (listof current-type nonunit)) 'BusDK)

(save '(idmt (listof impedance-type nonunit)) 'BusDK)

(save '(definite (listof current-type unit)) 'BusDK)

(save '(definite (listof impedance-type nonunit)) 'BusDK)

(save '(definite (listof current-type nonunit)) 'BusDK)

&& ***LINE Agent Knowledge Base*** &&

;;;; ***Unit Protection where there is communication link between two ends***

;;;; ***of the line.***

(save '(pilot-wire (unit comm-link)) 'LineDK)

(save '(current-diff (unit comm-link)) 'LineDK)

(save '(distance-comm-permissive (unit comm-link)) 'LineDK)

(save '(distance-comm-blocking (unit comm-link)) 'LineDK)

(save '(phase-comparison (unit comm-link)) 'LineDK)

;;;; ***Non-Unit Protection.***

(save '(overcurrent (nonunit comm-link)) 'LineDK)

(save '(overcurrent (nonunit no-link)) 'LineDK)

(save '(dist-impedance (nonunit comm-link)) 'LineDK)

(save '(dist-definite (nonunit comm-link)) 'LineDK)

(save '(dist-definite (nonunit no-link)) 'LineDK)

(save '(dist-inverse (nonunit comm-link)) 'LineDK)

(save '(dist-inverse (nonunit no-link)) 'LineDK)

For a complete list of the agents' knowledge base, refer to *Appendix C.*

7.3.4 Agent Functions

The agent's behaviour or functions are specified using performatives. In KQML, a performative is a term describing an operation or action. Agents use the expressive power of SIF (subset of KIF) to define their own performatives [Patil *et al.* 92, Smith and Poulter 93, Finin *et al.* 93, Finin *et al.* 94]. KQML also provides an extended list of predefined performatives, which deals with belief revision, queries, knowledge base maintenance, activities and services.

Performatives are defined using the macro *defperformative* which has the following form :

(defperformative <performative> ((receiver (eql '<agent-name>))
<arg1> <arg2> <arg3> .....)
(.....
body of performative
..... *)*

For example, the retrieve performative for the Bus agent is defined as :

(defperformative **retrieve** *((receiver (eql* **'bus**))
coordinatorcasekb buscasekb linecasekb)
(.....
body of performative
.....)

The performative arguments (i.e., coordinatorcasekb, buscasekb, and linecasekb) are called theories. These theories represent the case library in the agent knowledge bases.

Tables 7.1, 7.2 and 7.3 show the performatives for the Coordinator, Interface, and Bus and Line agents, together with a brief explanation of each performative. Table 7.4 lists the theories of the respective agents.

Table 7.1 Coordinator agent performatives.

Performative	**Definition**
decompose_problem	decomposes the original problem and assigns the tasks to the appropriate Design Agents.
retrieve_coordkb	attempts to retrieve coordinated designs from its memory.
check_design	checks whether coordination is required for new solution cases.
coordinate_design	coordinate the component protection schemes designed.

Table 7.2 Interface agent performatives.

Performative	**Definition**
run	starts communication with the user to design protection scheme for a power system or a part of it.
initialise_case	obtains initial information from the user with regards to the design of the protection scheme.
start_design	requests Coordinator Agent to start work on the user problem.
get_details	obtains component details from user.
case_details	passes component's details to the Design Agent which requests for it.
view_relay	allow the user to view the applicable relays related to a protection scheme.
display_design_case	shows the final design solution to user.
get_confirmation	gets approval from user as to whether design solution is acceptable.

Table 7.3 Bus / Line agent performatives.

Performative	**Definition**
start_design	starts designing for the new assigned problem case.
check_similar_cases	checks for existence of similar designs in individual's case memory.
new_design	designs a new protection scheme from first principles.
retrieve_similar_cases	retrieves similar design from case memory.

Table 7.4. Agent theories

Agent	**Theory**
COORDINATOR	coordinatorcaseKB (stored cases) and relayDK (stored KB)
BUS AGENT	buscaseKB (stored cases) and busDK (stored KB)
LINE AGENT	linecaseKB (stored cases) and lineDK (stored KB)

7.3.5 Communication among ISPP Agents

The multiagent system consists of a set of agents which communicate by means of knowledge-level coordination primitives defined according to the speech act theory [Austin 62, Searle 69, Grice 89, Bussmann and Miller 93]. The speech act theory provides a general framework for modelling human communication. The basic unit of communication among agents is the transfer of a message from one agent to another. The purpose is to provide the receiver of the message with some information or to have the receiver take certain actions [Patil *et al.* 92, Genesereth and Ketchpel 94, Jennings 94]. For example, a communication is initiated when an agent requires some data, information or assistance from another agent. In this instance, the agent sends a message and extends its request to the appropriate agent. The receiving agent of the message then response by taking necessary actions and executing the requested task or tasks.

For a survey of agent theories, architectures and languages, refer to [Wooldridge and Jennings 94].

The agent communication language used in the implementation of the prototype includes two main components :

- A representation language for the contents of messages, such as KIF or SIF which we use in our system [Genesereth and Fikes 92];
- Communication KQML which consists of a set of communication primitives called performatives, aims to support

cooperation among agents in distributed applications. The KQML performatives enable agents to exchange and request knowledge, and to cooperate during problem solving.

The agents in the system communicate by sending KQML packages. A package contains information about the sender, receiver, contents of the package, communication mode, etc. The content is described using SIF performative.

The general form of a package is given as :

(package :content
<performative or SIF description of an action>
:sender <agent-name>
:receiver <agent-name>
:reply-with <identifier>
:in-reply-to <identifier>
:commode <type of communication>)

The *:reply-with* field indicates whether the sender expects a reply to the package. And if the value of the *:in-reply-to* field is not nil, then the current package is a reply to a previous request. It is the responsibility of the receiver to send a reply if one is expected.

For example, when the Interface agent first receives a request from the user to design a protection system, it sends a message to the Coordinator agent. The following is an example of a message sent by the Initiator agent :

```
(package :content  '(start-design
                     buscasekb linecasekb coordinatorcasekb)
         :sender   'interface-agent
         :receiver 'coordinator)
```

where **start-design** is a message to Coordinator agent to commence the design process.

7.3.6 System Operation

The prototype system of ISPP implemented as a distributed knowledge base involves the bus and line components of a power system. The system contains some verified coordinated protection schemes in the case memory of the Coordinator agent and some verified bus and line protection schemes in the Bus and Line agents memory respectively.

The operation of the system begins with the user specification of the bus and/or the line to be protected (e.g., whether backup protections are required, the cts characteristics and so on). Provided with this information, the system goes through the communication process among the agents based on message passing paradigm. The result is a coordinated protection scheme which could be a verified design or a design which has been modified and adapted. If the solution is adapted from a similar retrieved design or has been built from scratch, then the solution represents a workable option which has not been verified yet. Verification of a case would require an expert's approval. From the engineering point of view, it is important to store cases which has been

subjected to thorough analysis and proper evaluation. Reliability represents a crucial factor in power system protection because of the risk factor involved and the undesirable consequences which would result if a fault occurs.

7.3.7 An Example of a Coordinated Design

The following example illustrates the design of protection schemes for a section of a power system. The system shown in Figure 7.4 consists of a bus and a line where backup protection is required. This sample program highlights the activities and communication between the agents.

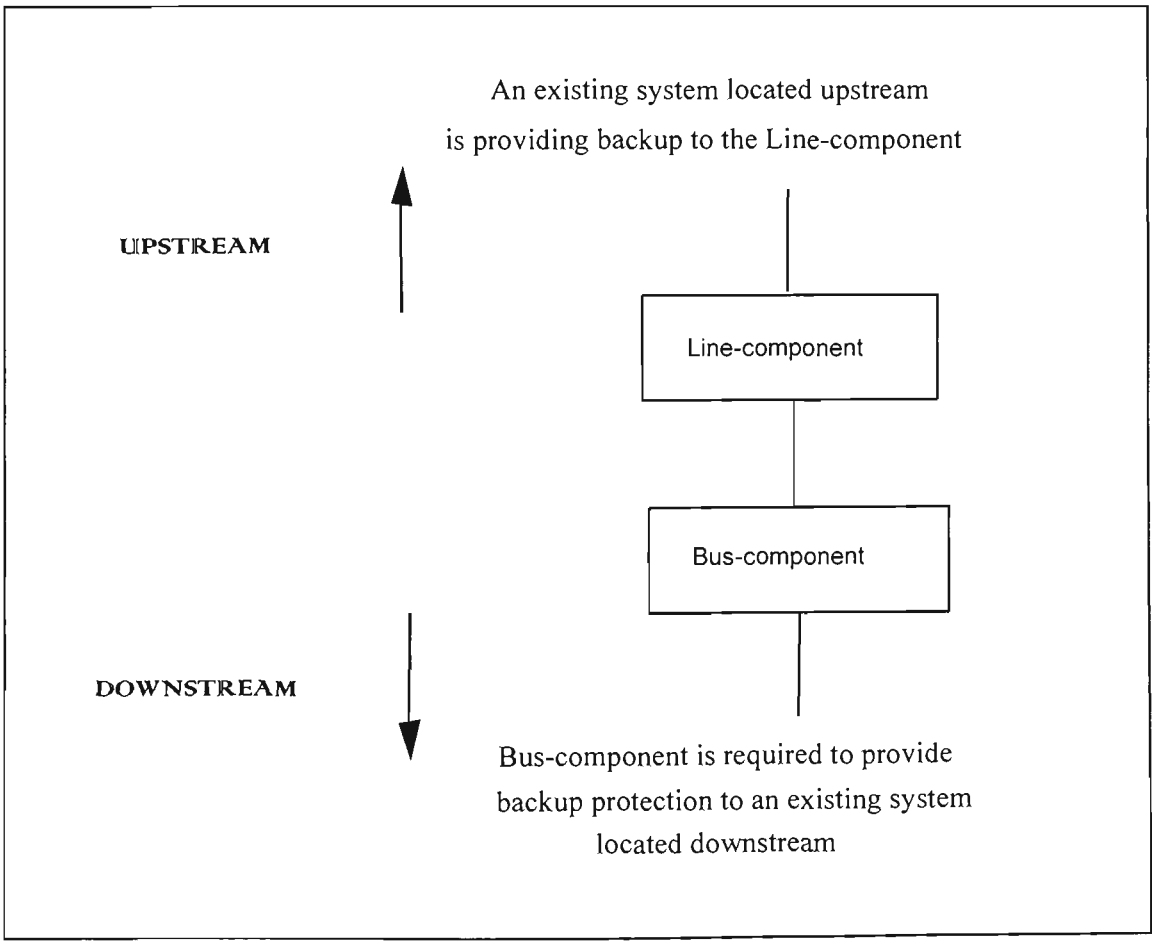


Figure 7.4 Components of a section of a power system requiring protection

The upstream direction refers to the direction where the source of electricity is located. The Line-component refers to a transmission or a distribution line of a power system. The Bus-component is a metallic conductor supported by insulators that interconnects the loads and the sources of electrical power in an electric power system.

At the beginning of each session, the user specifies the power system or part of the power system for which protection is required. The type of protection schemes applicable for each component depends on the component's position in the power system. If the system involves several components, the design will normally commence with the downstream component. Moreover, a protection system for multicomponent system will require coordination as well.

The complete example of the system operation is given in *Appendix D*.

7.4 Distributed / Shared Knowledge Base System in an Object Oriented Environment

The architecture of the second system has been implemented in an object oriented database system, O_2 [O_2 System 93]. O_2 is an object oriented database management system.

The system employs a Blackboard structure which forms an integral part of the system. The Blackboard is used as a medium for communication and as an integration platform for posting and coordinating partial solutions. Similar to the knowledge base system discussed in the previous section, this system also

uses distributed problem solving technique, and incorporates different reasoning methodologies such as rules, case based and explanation based reasoning. The development of the prototype has provided substantial support for the system architecture as a working model.

7.4.1 Objects / Classes

The application of object technology provides support for data abstraction, knowledge encapsulation, overloading, inheritance, reusability of software components, extensibility and modularity. Thus, applying object oriented paradigm has reduced the implementation effort and increased the efficiency of the program at the same time. Another attractive feature of object technology that makes it appealing to use is the possibility of allowing experienced users to customise a system according to their particular requirements. It enables the system programmer to define complex object types together with a set of operators which are specific to the application.

Implementation in an object oriented system is relatively easy. As everything in this world can be viewed as objects, each agent can also be viewed, designed and constructed as an object/class. As shown in Figures 6.9 and 6.10 in chapter six, there are three types of agents in the system and the agent architectures consist of similar components. Hence, a superclass can be constructed where all the agents can inherit from it. In other words, in the object oriented environment, the agents are the software components, which are represented as classes.

7.4.1.1 Constructing the Agents

The problem domain (i.e., protection for power system) can be divided into a number of smaller domains, each focusing on a particular part of the power system (e.g., busbars, lines, transformers, generators, and motors). Each Design agent defined in the system represents one component of the power system. Therefore, a Design agent maintains its own domain knowledge and inference engine and specialises in solving problems in a particular aspect of the problem domain. Examples of the Design agents in the system could be *Busbar, Lines, Transformers, Motors* or *Generators* and any other components of a power system that requires protection.

All the components represented by different Design agents have certain similar attributes. Thus, in order to avoid duplicating the code and to reduce the programming effort, a superclass can also be created where all the Design agent classes can *inherit* from it. Figure 7.5 shows the object hierarchy of the agent classes in the system.

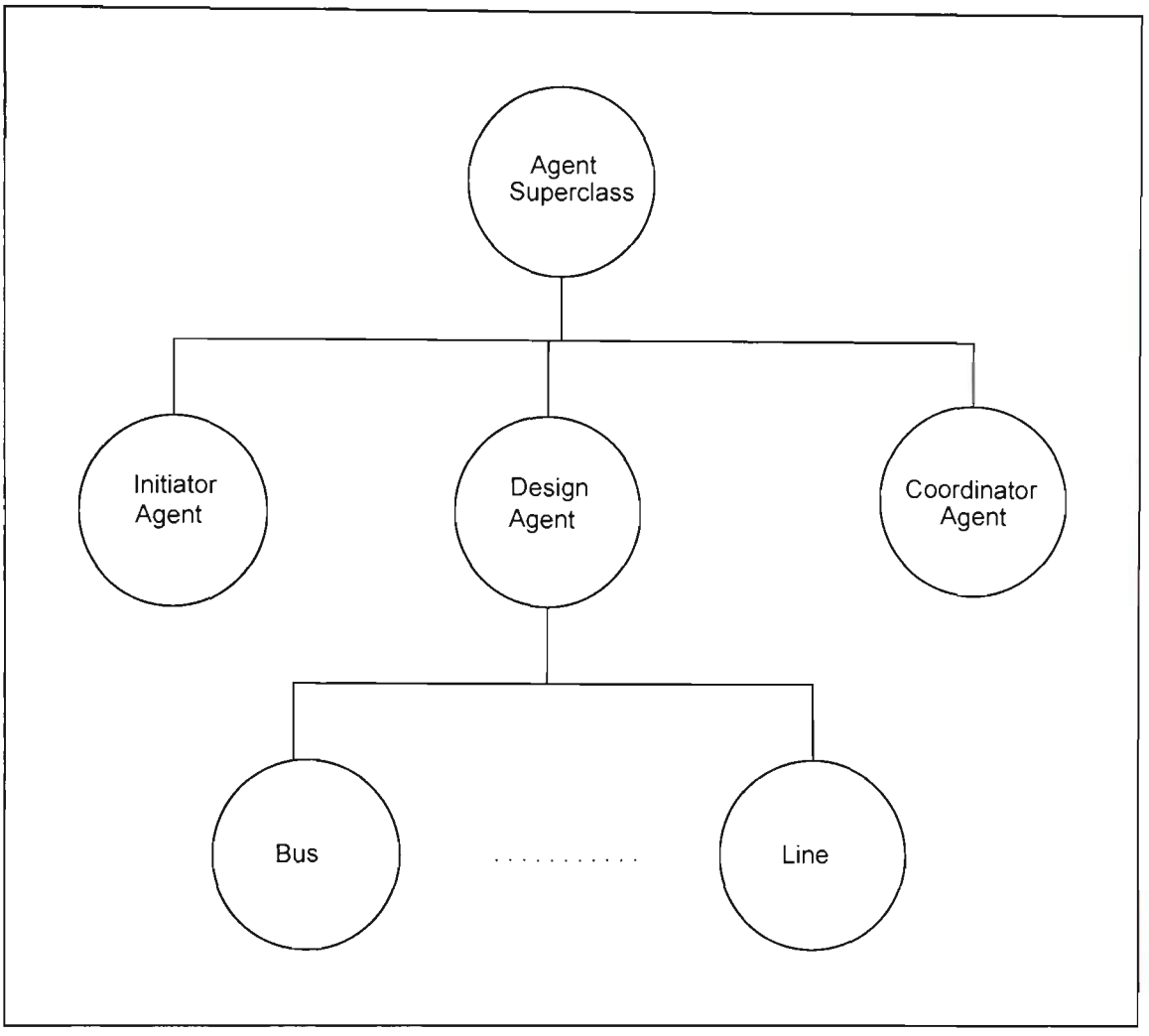


Figure 7.5 Objects/Classes in the System

Each agent class represents the state of the agent, and the methods represent the behaviour of the agent. The multiparadigm reasoning techniques are modelled and encapsulated into each class as well. Object technology provides the feature of encapsulating the knowledge and intelligence required by each agent into a class.

As mentioned earlier, the function of the Initiator agent is to initiate problem solving task in the system by decomposing the original problem and distributing the decomposed tasks to the various Design agents. The functions of the Design agent are to build a solution to the given sub-problem by applying and adapting

the solutions of similar cases from the case library or construct new designs from scratch. The Coordinator agent is basically responsible for coordinating the various solutions into one single integrated solution and providing an explanation as to how the solution was derived. The multiparadigm reasoning strategies employed by the agents are represented as class methods.

Both the *Case Library* and the Blackboard structure are also constructed as objects of classes. The Blackboard is a global memory area which is introduced into the system to facilitate problem solving, to record information and which serves as a work area for the Coordinator agent. The *Case Library* represents part of the system's distributed knowledge and forms the Design agent *expanding* experiences in solving domain problems. The *Case Library* keeps previous design cases which are stored as *persistent* objects in the database. *Persistence* is provision for values to remain computationally available for an arbitrary length of time; as long as they are required for computation [Atkinson *et al.* 93].

7.4.2 System Architecture

As stated earlier, the system's architecture consists of a group of loosely coupled and decentralised problem solving agents. This system architecture has been presented in [Wong and Kalam 95]. The system is a distributed knowledge based system where agents have to cooperate and combine their knowledge. The agents are organised in a hierarchical structure and employ multiparadigm reasoning strategy to solve their individual tasks. The system has three types of agents: Initiator, Coordinator and Design agents. The expertise required to solve

the domain problems is partitioned among the domain specific agents. Even though the agent architecture described here is very similar to the agent architecture described in the previous chapter, the system architecture is however, quite different.

The organisation of the system's agents is shown in Figure 6.6. The organisational structure of the agents determines the amount of information processed and the coordination necessary for the agents to operate efficiently [Shaw and Fox 93].

7.4.3 System Operation

The system is designed to communicate interactively with the user. The implementation of the prototype consists of the busbar and line components. Information required by the system to start designing a protection scheme include configurations and constraints of a busbar or a line, for example, information on the cts, requirements of a protection system, and specifications of the component, if necessary.

Given the required information, the system would first automatically attempt to build a solution by applying its experience in dealing with similar problems. However, if the current problem is new to the system, the will start to construct a new case from scratch. Coordination will be required if the protection system to be designed covers more than one component of the power system.

Figure 7.6 shows the operation of system and the interactions between the various agents.

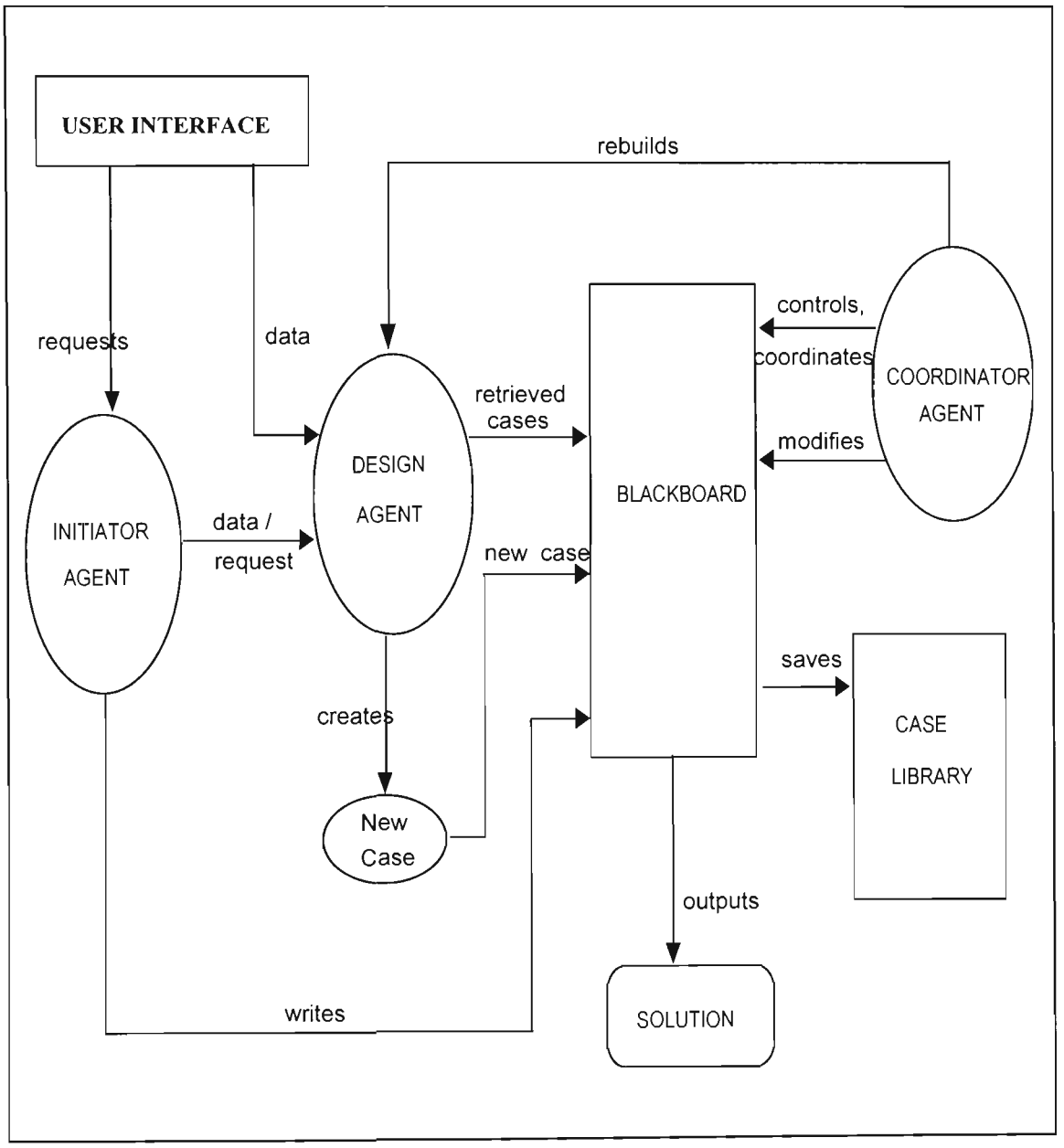


Figure 7.6 System Operation

The system commences the design process with the user requesting a solution (i.e., to build a protection scheme for a power system). This request is received by the Initiator agent which than decomposes the problem, if possible, and assigns the sub-tasks to the appropriate Design agents. Identity of these

Design agents is written on to the Blackboard. Each Design agent then starts its problem solving task and interacting with the user whenever necessary to obtain further data and information.

At the end of the process, a solution case is derived, either by adapting and modifying a similar case from the case library or creating a new solution case from first principles. As the solution case is posted to the Blackboard, the Design agent also updates the information in the Blackboard. When all the partial solution cases are posted to the Blackboard, the Coordinator agent then begins coordinating the cases to form a complete, integrated solution case. The Coordinator agent may also be required to provide answers to any queries the user may have.

From the system operation, a flowchart of the system which shows the control flow of the system can be derived. Figure 7.7 presents a flowchart of the system.

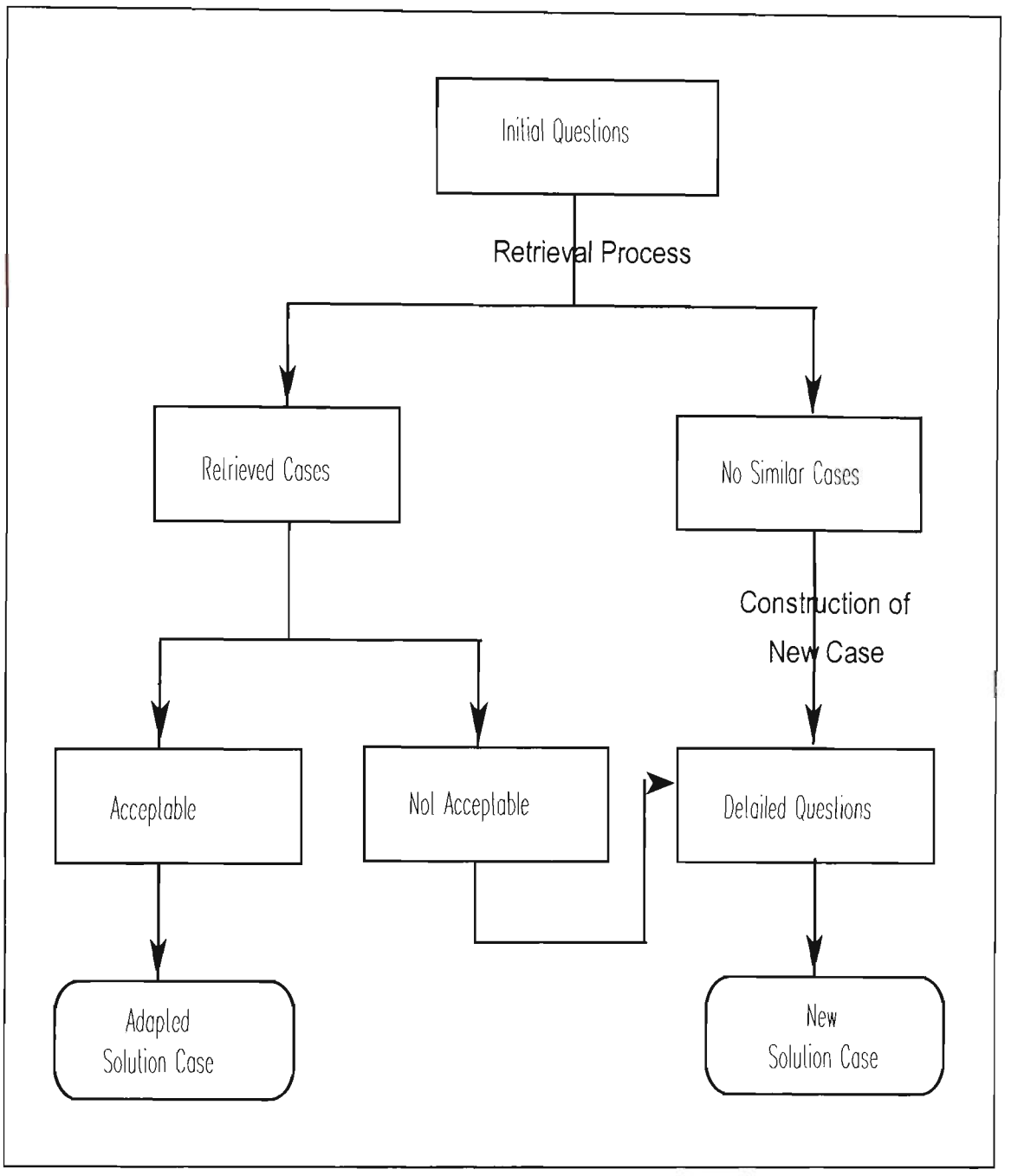


Figure 7.7 Flowchart of the System Operation

The flowchart diagram shows how the process of the formation of the adapted or new solution case is performed. During the retrieval process, the Design agent may retrieve one or more similar cases. When similar cases are found, they are presented to the user for confirmation of acceptance. This step could be automated. If accepted, the solutions of the retrieved case are adapted to the problem case.

However, if the retrieved cases are not acceptable or there were no similar case found, a new solution case has to be then constructed. In such a situation, the problem solving would have to begin from first principle basis. In the second instance, where the Design agent fails to retrieve any similar cases, problem solving would also begin from first principle basis.

Given the flowchart, the following illustrates the working example of a problem posed to the system.

7.4.4 Illustration of a Design Problem

Consider for example, the situation where the user wants to design a protection system for a busbar. The Initiator agent initiates the problem solving by communicating with the user to collect the relevant data. Given the information on the current transformers (cts), requirements for back-up protection and so on, the Initiator agent then decomposes the problem into smaller tasks and distributes them to the appropriate Design agent. In this example, the task is assigned to the Busbar agent. The identity of the Busbar agent is also written onto the Blackboard to let other agents know which Design agents are currently involved in the problem solving tasks.

The Busbar agent, equipped with the data and information will search its *Case Library* for similar designs. If found, the solutions of the retrieved designs will be applied and adapted to the given busbar configurations and presented to the user as the possible design solutions.

If the Busbar agent has no experience in solving the busbar problem at hand, it will interact with the user directly and request for more information. The Busbar agent applies this information together with its built-in knowledge to derive a new design solution from first principles. In addition to the built-in knowledge, the Busbar agent may also access to the Shared Domain Knowledge which contains facts about the domain. Examples of some of these facts included in the Shared Domain Knowledge are the related information provided by the different manufacturers about the type of relays available in the market place.

When the Busbar agent completes its task, it posts the new or adapted design to the Blackboard and updates the information on the Blackboard. Consequently, all design cases representing the partial solutions constructed by the Design agents involved in the current phase of problem solving, are posted to the Blackboard. While the Initiator agent records initial information regarding the current problem, updates are done by the Design agents as they complete their tasks on the Blackboard. Once the Busbar agent has completed and posted its design to the Blackboard, the Coordinator takes over and proceeds with the coordination work.

The Coordinator agent looks at all the partial solutions on the Blackboard and attempts to coordinate them into an integrated and complete solution. In this example, where integration with other partial solutions is not required, the busbar design is presented to the user as the final solution.

7.5 Distributed Knowledge Base System within a Federated System

This section discusses the implementation of ISPP as a distributed knowledge base system using a federated system structure in an object oriented environment. The agents here are cooperative and distributed, similar to the agents in previous systems. However, in addition to that, the agents surrender their autonomy to a third entity (i.e., a local Facilitator). All agents of the same type are gathered in one community, governed by a Facilitator. Hence, there is no inter- or intra-communication allowed among the agents and communication is only achieved through Facilitators.

The design and selection of a protection scheme for various parts of the power system requires not only domain knowledge but experience and skill as well. An expert who is good in analogical reasoning may not be good in remembering. As stated before, whenever a problem is encountered, it is quite natural for humans to investigate past problems and try to either adopt or adapt the previous solutions to the current situation. Therefore, case based reasoning paradigm appears to be the most appropriate reasoning technique to be adopted for the development of the protection system.

The representation of the design cases using object oriented methodology is not only neatly modelled but the features and behaviour associated with the agents (also defined as cases) can be easily described. The agents are grouped together to form communities (also known as case libraries) and are stored in the object oriented database. The development of the system involved numerous discussions and interviews with the protection engineers. The knowledge and

the expertise acquired during these discussions form the heart of the system and are used to construct the knowledge bases.

7.5.1 System Development

The system has been developed in O_2 [O_2System 93]. O_2 is an object oriented database management system (OODBMS) based on C language. It comes with a complete development environment. Two components covering the busbar and line protections for a power system have been incorporated. The agents created are kept in communities, similar to the way previous design cases are kept in the case library.

The flow of system control and the interactions of the system's modules is shown in Figure 7.8.

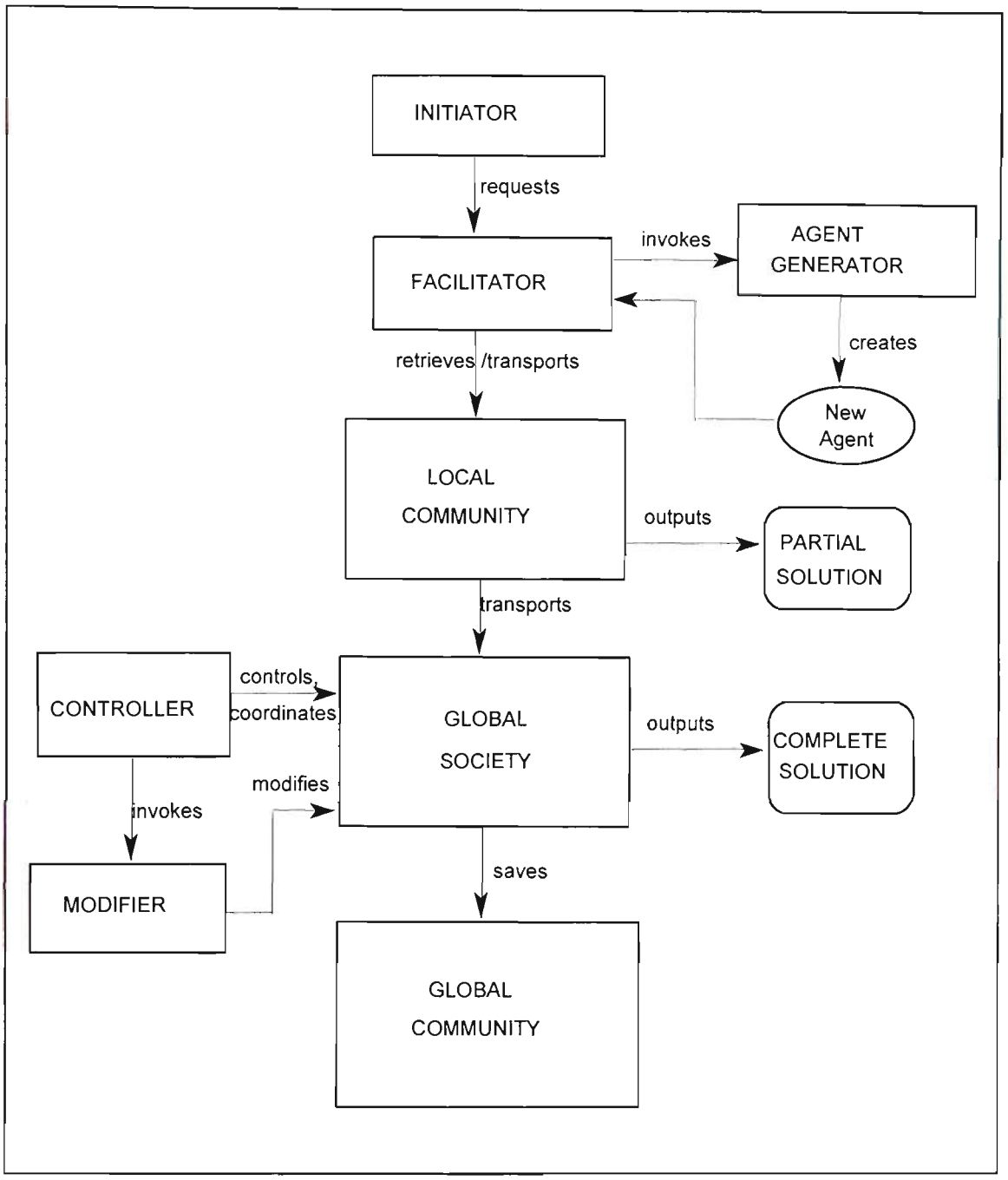


Figure 7.8 Interactions between System Modules

Given an initial set of specifications for a busbar, say, the Initiator module invokes the appropriate Facilitator. Applying the case based reasoning technique means that the Facilitator would use the set of specifications to construct a solution. It first attempts to recall the busbar agents from the Local Community that match the requirements. These agents, if found, are then transported to the Global Society. However, if no agent responds to the

Facilitator's message, it means that case based reasoning has failed. In this instance, the Facilitator will invoke the Agent-Generator module to generate a new agent.

The new agent is constructed to meet the design specifications and functional requirements for busbar protection. This new agent, once created, has the intelligence to derive a solution to solve the design problem associated with the busbar. It also has the capability to explain how the solution has been derived. Based on some of the new agent's attributes, the Facilitator will try for the second time, to retrieve any similar agents from the Local Community. The purpose of a second retrieval is to ensure that no duplication of agents would exist in one community. However, even though a similar agent may be found during retrieval time, the user may disagree with the solution, thereby causing a new agent to be generated.

If the system needs to design a protection scheme which covers several parts of a power system (e.g., the busbar and lines connected to it), then some sort of coordination would be required. This is achieved by interacting and coordinating the 'retrieved' or new agents which have been posted to the Global Society by the Controller. In the process of coordinating the agents, the Controller may find that some modifications and adjustments are necessary to some of the agents. When this happens, the Controller may invoke the Modifier to perform its responsibility to adapt some of the agents so that they may be coordinated with one another. The result of this coordination process is an integrated and coherent protection design scheme for a power system.

7.5.2 System Operation

The system operation is illustrated in Figure 7.9, which shows part of the system which has been implemented.

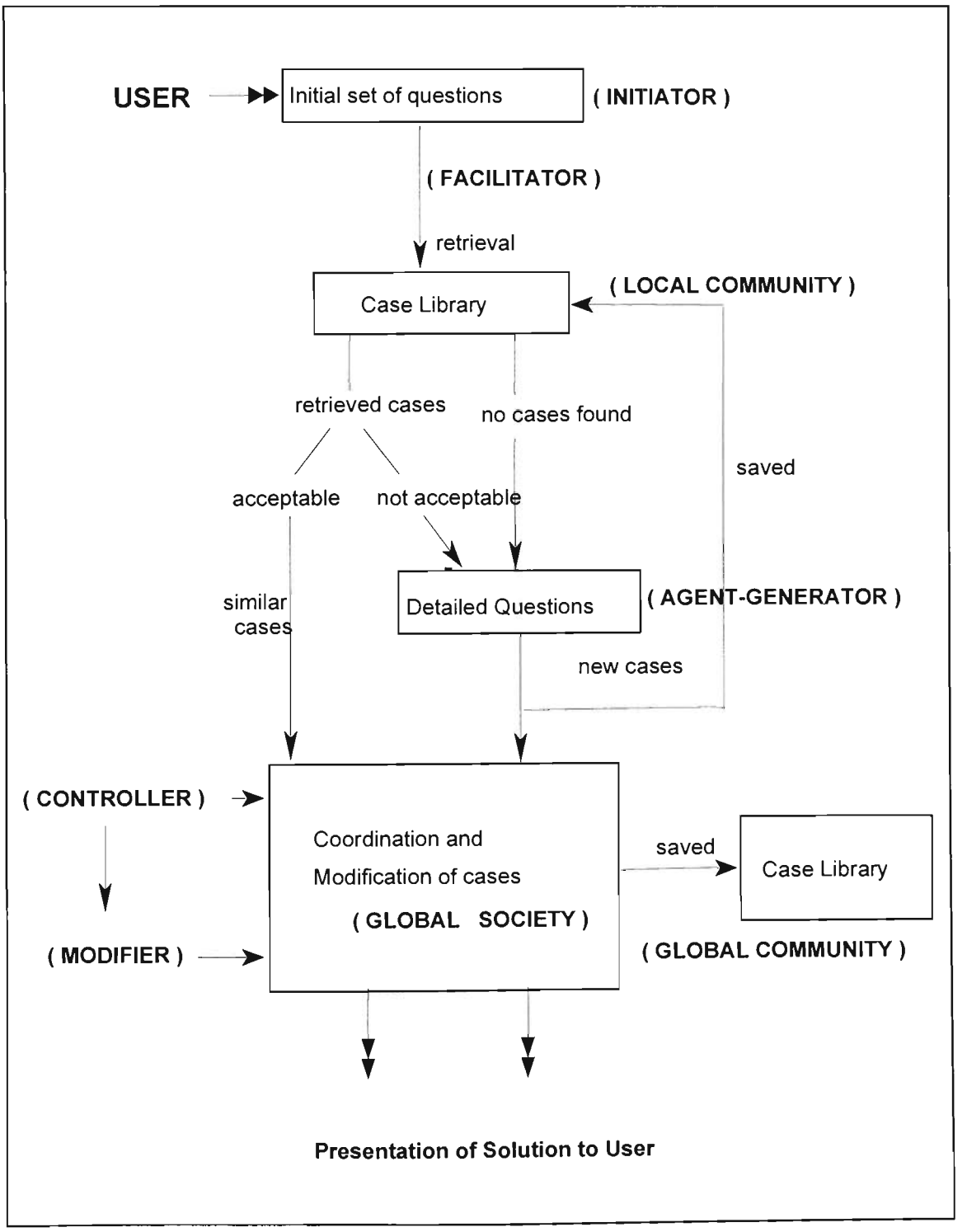


Figure 7.9 System Operation

The operation of the system begins with the Initiator asking the user some fundamental questions regarding the component of the power system to be protected. Sample questions asked are whether remote protections are required, whether dedicated current transformers are available and their characteristics. The Initiator then passes the user's specifications to the appropriate Facilitator(s). An attempt is then made by the Facilitator(s) to retrieve similar cases (represented as agents) from the case library (represented as Local Community). If no suitable cases are found, the Agent-Generator is invoked to construct a new case, based on the user's specifications.

The retrieval process is then repeated to ensure that the newly created case does not exist at all. If it does not, the case will be added to the case library. The retrieved cases or the new case are later transported to the Global Society where the coordination and modification processes may take place, if necessary. The result generated by the above procedure represents the solution which is then presented to the user and saved into the Global Community.

The system operation just presented is illustrated in a flow chart shown in Figure 7.10.

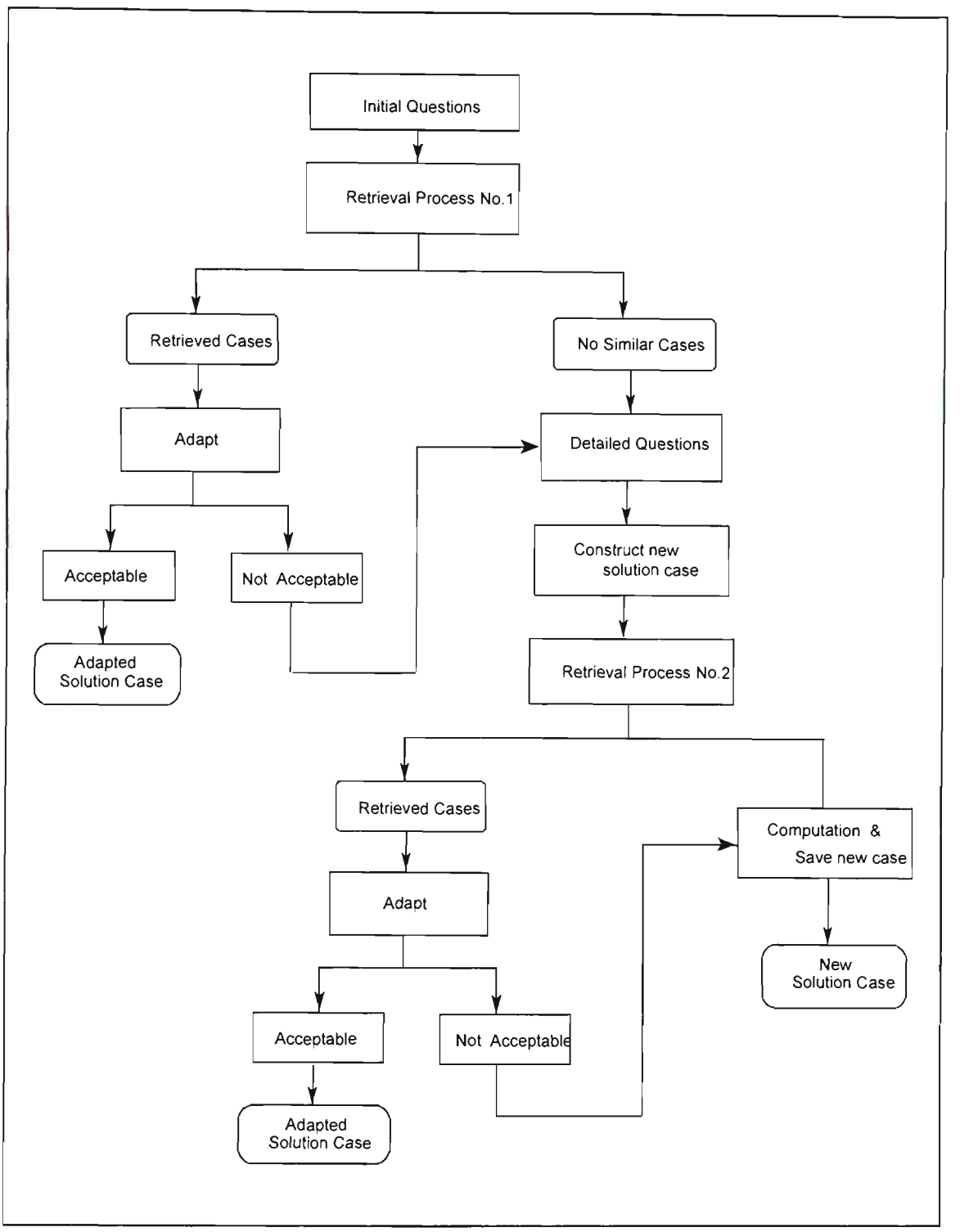


Figure 7.10 Flowchart of the System Operation

The flow chart indicates very clearly how the system functions during problem solving. Note that some of the repetitive processes (e.g., retrieval and adaptation processes) can be accomplished easily through reusability of code as provided by object oriented programming.

7.6 Conclusion

This chapter discussed the implementation and the operation of the ISPP system, which has been developed using three different architectures embedded in different environments and under different operating constraints. The three systems are multiagent systems which have been developed applied to power system protection - an application area which is representative of an important class of engineering design problems. The systems employ intelligent agents that interface with the external system (which could be a human or another system), to solve partial problems and produce partial solutions, and to perform coordination task.

However, the main focus of this chapter was to reinforce the investigation carried out in the previous chapter regarding how the autonomy of the agent affects and generates different system architecture. The three systems have different organisation of agents and possibly employ additional modules or Blackboard structure depending on the development environment, system requirements and constraints. Nevertheless, the agents employed in the three systems have similar attributes. The components within the agents have been implemented as layers because layering provides a powerful technique for organising the components of an agent. Some of the agents have to be modified and adapted from the generic agent architecture that was designed and developed as illustrated in chapters four and five. This shows the adaptiveness of the generic agent architecture which allows the incremental development of other types of agents, modules and eventually the whole system itself regardless of the environment, operating system or constraints.

Chapter Eight

SYSTEMS EVALUATION

Abstract

This chapter discusses the advantages and limitations of the three architectures which have been developed and prototyped as illustrated in the previous chapter. All three architectures have been employed to develop an intelligent multiagent system in power system protection (ISPP). The multiagent system is used to design protection schemes for a power system. The architectures have similar attributes and employ similar type of agents. The construction of the agents in the multiagent system originates from the same agent architecture as discussed in chapter five. This agent architecture that we have introduced in chapters four and five is a generic architecture which could be used to build intelligent and adaptive agents for the development of intelligent multiagent systems in various application domains. In addition to that, the systems employed distributed problem solving technique and integrated different reasoning methodologies such as case based, rule based and explanation based. However, the dissimilarities between the systems began with the environment in which they have been developed, the organisation of the agents in the system, the redefinition of the agents' responsibilities, the mode of communication used, the employment of additional modules and the degree of autonomy held by each agent in the respective system.

8.0 Introduction

System design has never been an easy task; it requires skills which are central to many human tasks and which are used in many professions. Design is a complicated activity and is always tied to some constraints and relies not only

on knowledge of design styles and domain knowledge such as principles and performance theoretical guidance, but also frequent references to previous designs. According to Chandra [Chandra 92]:

> *"Design is the process of producing artifacts that have desired properties and meet a set of functional requirements."*

Whether produced by architects or by protection engineers, the designs must have certain required properties which fulfil a set of functional requirements. However, if the design problem is ill-structured, then design experiences and heuristics play a very important role in the design process. Most designers rely on prior designs and draw upon their knowledge to generate new designs and solve current problems.

In practice, designers always refer to past experiences and even the work of other designers when they are working on new design problems. This feature is embedded into ISPP so that the multiagent system will imitate the human expert's natural approach to problem solving. This means that the multiagent system has the intelligence and capability to be able to relate to past experiences to when solving the current problem. The multiagent system architectures presented in the previous chapter, all share a number of similar characteristics - they all utilise distributed problem solving technique as well as a multiparadigm reasoning strategy when solving problems. Furthermore, they employed three main types of agents in the organisation of the system: Initiator agent, Design agent and Coordinator agent.

8.1 The Multiagent Systems - A General Perspective

Today there is a growing demand for decision support systems which can interoperate with other systems and applications. However, as the systems grow in complexity and size, they become more difficult to maintain and almost impossible to change. The multiagent paradigm introduced in the ISPP presents a novel approach to power system protection design. Multiagent approach promotes the development and utilisation of smaller and more manageable system components. In addition, agent technology provides a framework in which new and existing (heterogeneous) components can cooperate to achieve a common objective.

Instead of building a single monolithic expert system, a multiagent architecture consists of several distributed expert systems that can communicate and share knowledge. The advantages of this methodology are twofold:

(i) smaller components are simpler and more reliable because of reduction in complexity;
(ii) system decomposition aids the problem of conceptualisation and increases the system modularity, thus making the system more manageable and easier to understand.

The system's knowledge is distributed among the different agents. Each agent specialises in a particular subset of the domain area. This arrangement enables a particular agent's knowledge to be updated without affecting other agents or components. Moreover, additional agents can also be introduced relatively easily into the system. However, this is not true with traditional expert

systems where changes to some rules will effect the system knowledge base. Furthermore, introducing additional components or subsystems may result in revamping the entire system.

One of the main advantages of this architecture is the increased modularity due to distributed control and the integrated approach, which makes the system more manageable and easier to maintain. The employment of different agents to carry out domain specific tasks enables complex problems to be solved more effectively and efficiently. The approach shows how distributed knowledge base can be integrated and how the multiparadigm reasoning strategy can be employed.

Another advantage of the agent based approach is the ability of the agents to employ multireasoning strategies ranging from rule based reasoning through argumentation and case based reasoning. The agent's inference mechanism must be sufficiently advanced to reflect the expert's way of thinking. The ability to use multiple reasoning strategies, selecting the most appropriate strategy for the task while switching between the strategies when necessary, appears central to the success of expert reasoning. Majority of experts, including protection design experts, would normally rely on their experience when solving problems. If they have not been confronted a similar problem before, they would apply their domain knowledge and arrive at a solution using first principles. This strategy has been implemented in the ISPP.

One solution proposed to overcome the complexity barrier is to build systems of smaller and more manageable components which can communicate and cooperate [Genesereth and Ketchpel 94]. A distributed system which

consists of a community of communicating and cooperating agents is far more flexible, versatile and modular compared to the older generation of expert systems. The new generation systems can be tested and expanded incrementally with minimal changes to the existing system. In addition, the knowledge which is contained in the system can be fully or partially reused when adding new system components.

The employment of multiparadigm reasoning strategies allows the system not only to choose the appropriate strategy but also to switch between the reasoning paradigms, thus mimicking the behaviour of a human expert. Moreover, the application of multiple and distributed expert systems offers more flexibility, integrity, robustness and many other advantages over and above the conventional approach which involves one single large expert system.

Case based reasoning is employed as the main reasoner of the system. It utilises specific knowledge of previously experienced concrete problem situations to solve the current problem. The case based framework provides a rich experience-based environment where problem solving does not have to start from first principle, thus reducing time, cost and effort in the formation of a solution. The system is also capable of learning and this increases its capacity to reason and solve new problems through the expansion of its case library.

Last but not least, the implementation of different reasoning paradigms and switching between them, knowledge sharing, incremental growth and changes to the knowledge base can be achieved more easily and naturally when implemented in a knowledge base system. It would be difficult, if not

impossible, to build in similar versatility and adaptability into the system components using traditional tools and approaches.

8.2 Evaluation by Comparison

The three system architectures of ISPP have similar as well as dissimilar features. As discussed in the earlier section, the similar features include the use of agent technology, multiparadigm approach and the employment of case based reasoning as the main reasoner in the system.

All three architectures incorporate a multiparadigm approach and are implemented using a layered technique. The layered architecture provides an attractive and powerful way of representing and integrating various paradigms and components into a single architecture.

One of the similar features includes the use of multiparadigm approach with case based reasoning technique as the main reasoner in the system. Multiparadigm reasonings strategy is employed to enrich the system's reasoning concepts and increase its reasoning capacity. This means that different reasoning techniques are applied at different times during the problem solving process depending on the situation at hand.

The systems employ multiple representation of knowledge. Domain knowledge is maintained in the form of previous experiences (as in the case library) as well as in the form of rules and facts. Rules and facts are utilised to derive solutions from scratch. However, the knowledge is represented quite

differently in the three systems. For example, in the original architecture, the experiences are kept in the form of integrated cases in the agent memory whereas, in the federated system, the cases are represented as agents residing in a community.

The amalgamation of the domain knowledge with past experiences enhances the system problem solving technique capability. Moreover, the application of distributed problem solving encourages modularity and increases the speed of problem solving. This is accomplished by decomposing a problem and distributing the tasks to the various problem solving agents instead of having just one expert system to perform all the various tasks in the entire system. Furthermore, with the decomposition of complex problems into smaller sub-problems, distributed problem solving enables more complex problems to be solved.

The application of case based reasoning paradigm has resulted in the systems which are more competent, useful and intelligent. This is brought about by the dynamic expansion of the system's experiences in problem solving. In addition, case based reasoning can also handle incomplete information or missing data quite well, that is, the feature of the missing values will not be used in the retrieval process. Each case is indexed on a number of salient features and the missing feature(s) can determine the importance of the case as regards the proposed solution. If similar cases can still be retrieved, then the missing values have proved to be unimportant. Otherwise, another reasoning technique can be applied to produce a new solution.

The three systems differ in the overall agent architecture and the organisation of the system with a varying degree of distributed control and processing, and also on the introduction of a number of entities or components in one system or the other. The systems employ different agent organisations where the agent responsibilities and the degree of the agent's autonomy differs for each system. In addition, there are two types of environments which have been used to develop the systems - knowledge base environment (Lisp) and object oriented environment (O_2). These diversifications and differences between the three systems have contributed to the advantages and limitations of each system.

However, it must be noted that evaluations must be made on the system architectures and that these evaluations can be performed by comparing the differences between the three systems. Therefore, it can be concluded that the dissimilarity features between the systems form the basis for these evaluations.

8.2.1 The Distributed Knowledge Base System in Lisp

This system uses message passing as the mode of communication. Message passing also offers a flexible, efficient and sophisticated mode of communication either between the layers within an agent or among the agents in the system. All communications with the external source would have to be initiated by the interface agent. The result of this form of communication is a well managed flow of messages and interactions between the agents in the system. On the other hand, if the communication flows are not well managed, it

could also easily brings about a bottleneck situation with the interface agent being loaded with too many requests to communicate with the external source.

Apart from the agents already mentioned, there are no other modules or entities employed in the system. Thus the management of the system is made easier as the system is controlled and run by three types of agents only - interface agent, problem solving agent and coordinating agent. The agent technology and distributed problem solving technique allow the system knowledge to be modelled and distributed among the different agents, thus promoting the specialist agents in the system. This type of agents specialise in a particular subset of the domain area so that individual agent's knowledge can be updated without affecting other agents or components in the system. The agents which are also known as semi-autonomous agents have the capability of solving problems in their specialised domain area and producing partial solutions.

The implementation of a knowledge base system in a knowledge base environment is relatively easy, and therefore, the construction of each agent is made simpler. The agents' knowledge is built in the form of sentences or lists. Each agent inference mechanism is sufficiently advanced to reflect the expert's way of thinking. It has been noted that the ability to use multiple reasoning strategies, selecting the most appropriate strategy for the task while switching between the strategies when necessary, plays a central role when assessing the success of the system.

8.2.2 The Distributed/Shared Knowledge Base System in O_2

The utilisation of object oriented technology can also greatly assist in the construction of a multiagent system. The system can be readily implemented with the mechanisms offered by object orientation such as software reusability, inheritance, encapsulation, polymorphism and modularity. These features allow the system to be easily modified or expanded.

In this system architecture, common knowledge is shared by the agents. The purpose of using a shared domain knowledge is to increase the effectiveness of knowledge sharing and promotes efficiency of the system. At the same time, this feature also encourages reusability and avoids the duplication of the knowledge in each agent. On the other hand, problems need to be resolved and decisions need to be made with regard to the managing of the shared component as well as also deciding who should be responsible for updating the knowledge component.

Furthermore, a Blackboard structure is employed as another alternative mode of communication between the agents in the system. The Blackboard architecture increases the agent's flexibility to interact with other agents. Messages can be posted to the Blackboard which is accessible by all agents. However, in message passing, only the receiver receives the message transmitted. While the latter method may result in increase of communication traffic in the system, the use of Blackboard may help to reduce the amount of messages routing in the system. However, this cost must be balanced with the cost of having a Blackboard which would include the memory requirement and management of the Blackboard.

The system architecture also permits all the agents to communicate directly with the user. Direct communication between the agents and the user, ensures that the traffic flow of communication traffic is kept at a low level. This is because direct interactions with the user do not require messages to be passed to an interface agent, resulting in the reduction of the traffic in the system. However, an interface layer would be necessary in each agent leading to a slightly more complicated agent architecture. Furthermore, the responsibility of the agent would also increase because the agent would be responsible for message interpretation and/or translation, and management of the interface layer. The interface layer, together with the interpretation and translation functions would then be duplicated in all agents, causing the problem of redundancy to arise in the system.

8.2.3 The Distributed Knowledge Base System within a Federated System

As well as having the three main types of agents already discussed, the organisation of a federated system includes several other entities which coexist in the system including an Initiator, several Facilitators, a Controller, several Agent-Generators and Modifiers. In addition, the system also includes local and global communities of agents.

One of the main features of the federated architecture which differentiates it from the previous two architectures is that the agents surrender their autonomy to the local Facilitator. In such a system, the agents do not have sufficient control or autonomy to work independently, instead they cooperate via the local

Facilitator. The disadvantage of this feature is that such cooperation could put a heavy load on the communication channel to the Facilitator. However, the communication traffic between the agents would be greatly reduced and any unforseen conflicts that may arise could be avoided.

The overall control of the system rests with the Controller. The Controller acts as the manager of the global community. This organisation helps to improve the coordination of all the agents in the system. Furthermore, the management of the system becomes more organised and methodical.

The use of agents encourage reusability of problem solving components. Employing object oriented technology enables the system to take advantage of the benefits of object oriented programming. For example, the representation of a case or an agent can be modelled easily and efficiently by encapsulating its attributes and behaviour into a single object. Hence, building the *Knowledge Base* and the *Inference Engine* becomes simpler. In addition, the object oriented database offers a natural environment for the construction of persistent objects which represent the cases in the system.

The *Solution Base* in the agent architecture stores a history of all cases or experiences - making no distinctions between successful and failed cases. The disadvantage of this architecture is that it could take a much longer time to retrieve a similar case from the *Solution Base*. In addition to that, the system would not be able to distinguish the degree of reliability of a case when presenting the solution to the user.

However, there is no distinctive part or appropriate feature in the system which could take care of the community growth. Therefore, the inflexibility of the system to provide for expansion of the community of agents could lead to the problem of over-population in the local as well as in the global community.

The existence of a number of entities or modules in the system could introduce a problem of system adaptation as it expands. As there exists a considerable high degree of cohesion in the system, addition of any new entities to the system could involve tedious reorganising of the current entities.

Based on the discussions above, a summary of the features, advantages and disadvantages of each of the system is given in the following Table 8.1.

Table 8.1 Evaluation of Systems by Comparison

	Distributed Knowledge Base	*Distributed/Shared Knowledge Base*	*Distributed Knowledge Base within a Federated System*
Features	The multiagent system is implemented in a knowledge base environment consisting of three types of semi-autonomous agents (i.e., Interface, Coordinating and Problem Solving agents). Communication among agents is achieved via message passing paradigm.	The system is implemented in an OODBMS[1]. There are three types of agents in the system (i.e., Initiator, Coordinator and Design agents). Communication among the agents is achieved via message passing and blackboard system.	The system is supported by other modules (i.e., Initiator, Facilitators, Controller, Agent-Generators and Modifier). The agents reside in a local and global communities where they surrender their autonomy to their Facilitators.
Advantages	- Easy management and expansion of the knowledge base. - Separation of the memory into primary and secondary sections enables efficient and fast retrieval of cases.	- Benefits of object oriented technology (i.e., reusability, modularity, encapsulation). - Knowledge sharing reduces the need for duplication and redundancy of data. - Direct communication reduces the traffic flow of communication with an interface agent.	- Benefits of object oriented technology (i.e., reusability, modularity, encapsulation). - No direct communication among agents - thus reducing unforseen conflicts that may arise.
Disadvantages	- Potential bottlenecks may occur where Interface agent could be overloaded with too many requests at one time.	- Use of blackboard may increase memory requirements and system management. - Requirements of central scheduling and control of the blackboard to maintain global consistency.	- Inflexibility of the system to expand the agents communities. - Potential problem of over-population of agents. - Problem of system adaptation due to existence of high cohesion in the system.

[1] Object Oriented Database Management System

8.3 Conclusion

The assessment to the development of a practical system that is, ISPP for the design of power system protection schemes has been presented in this chapter. Even though there are a number of ways of designing and implementing the system architecture, the overall approach adopted in each of the three designs is quite similar. Each system contains representation of the distributed knowledge in multiple forms and the amalgamation of the domain knowledge with past experiences which enhances the system problem solving capability.

The ISPP consists of several distributed expert systems, known as agents, instead of a single expert system in the traditional systems. Each agent is an expert in solving problems in its own specialised domain. The knowledge base is distributed and represented in different forms. For example, the Design agent has knowledge in the form of experiences (case library) as well as in the form of rules and facts in its domain knowledge. The rules and facts are utilised to derive solutions from scratch. Multiparadigm reasonings strategy is employed to enrich the system's reasoning concepts and increase its reasoning capacity. This means that different reasoning techniques are applied at different times depending on the situation.

It should also be mentioned that case based reasoning forms a major reasoner in the system. It cannot be denied that the paradigm has brought a number of advantages. The utilisation of the paradigm has encouraged reusability of knowledge and promoted learning through the expansion of the case library. The employment of the paradigm also enhanced the performance of the system, which compared favourably to other approaches. Even though the

current situation may move out of the system's range of experience, degradation of the system will be graceful and temporary only. This is because it 'remembers' the new cases and stores them in the case library for future retrieval, thus improving the performance once again.

The application of case based reasoning paradigm has also made the Design agents more competent, useful and intelligent through the dynamic expansion of their experiences in problem solving. Furthermore, case based reasoning is able to handle incomplete information or missing data in such a way that the missing values will not be used in the retrieval process. Each case is indexed on a number of salient features and the missing feature(s) can determine its importance regarding the proposed solution. If similar cases can still be retrieved, then the missing values can be considered to be unimportant. Otherwise, another reasoning technique can be applied to produce a new solution.

Moreover, the application of distributed problem solving encourages modularity and increases the speed of problem solving. This is accomplished by decomposing a problem and distributing the tasks to the various problem solving agents. Furthermore, distributed problem solving also enables more complex problems to be solved.

Furthermore, the implementation in Lucid Common Lisp 4.1 was facilitated using the additional two libraries - API and Epilog loaded into the subroutines of Lucid Common Lisp. The agents' definitions, its responsibilities and the communications between the agents were well-defined and well-managed. However, the implementation in O_2 was also relatively easy. The conceptual

understanding of agents and cases was enhanced with the aid of objects and classes. The reusability mechanism has also helped to reduce the coding and programming phase in the development and implementation of the systems.

There was a clear difference in the system which was implemented using the federated system framework. The agents surrendered their autonomy to the respective local Facilitator. This feature disadvantaged the system because it increased the agents dependency when they perform their tasks and carry out their responsibilities to solve problems in their domain.

While both the first and second architectures have similar advantages in their respective ways, the latter can be concluded to be more superior than the former architecture. The second architecture has a shared Domain Knowledge which made the system more appealing and attractive. It also employed a Blackboard that allows indirect communication to be carried out and messages to reach all agents simultaneously.

However, as a conclusion, there is no one perfect system architecture design for the implementation of a system in any application domain. There are always advantages and limitations of one system architecture compared to another. This is due to the way the agents or entities are organised and how their responsibilities are defined. However, proper study must be carried out on each system architecture design. The advantages and limitations of each system architecture must be given proper thought and considerations in regards to the system requirements, operating environment and constraints and other influencing factors before a final decision should be made to select the most

suitable system architecture for the implementation and development of any system in any application domain.

PART IV

THE CONCLUSION

Chapter Nine

CONCLUSION

Abstract

This chapter concludes the thesis with a discussion of the multiagent system, its applications and the contributions of the research. The merits of the adaptive approach in developing intelligent agents are presented. Further evaluations are drawn with the multiple organisation of the agents in the form of different system architectures. The comparisons between the systems conclude with a preference to one of the architecture underlying it. A review of previous various chapters is also given. This chapter closes with suggestions of future work and also future investigations in the area of power system protection.

9.0 Introduction - Review of Thesis

The main interest and concern of the research is expressed in chapter one. The chapter presented an introduction to the existing systems that are being used, the current technologies applied to the development of application systems and the future trends that follow on especially in power systems. The problem has been identified after a survey showed that while intelligent systems are expanding their boundary to power systems, the protection area still lacks attention. At the same time, there is not enough research projects being carried out concerning the architecture of the systems, an area which is slowly being accepted and recognised as a potential area of research in the near future. These revelations formed the main interest of the research and lead to a detailed examination of the nature of the research, which was then followed by the listing of the objectives and contributions of the thesis.

The literature review in chapter two indicated that the application of expert or intelligent systems to power systems has grown significantly to become an area of strong research interest especially in the past few years. Further survey revealed that there is an increasing interest in an attempt to employ and integrate AI techniques into newly built systems. This is indicated very clearly by the shift or swing in the recent developments and research where the more intelligent systems such as distributed knowledge base systems and decision support systems which utilise agent technology and incorporate fuzzy logic, genetic algorithm are replacing the more traditional systems such as rule based systems and neural networks.

The introduction to the study of architecture was presented in chapter three. It outlined the meaning and the nature of architecture study and at the same time, discussed the importance of conducting such study. In fact, the chapter serves as an introductory chapter and gave a review on the research work that has been carried out in the area so far.

The main focus of this research is on the study and design of an architecture for an intelligent system. After the initial review in chapter three, chapter four continued the discussion with the presentation of a new adaptive approach to the design and construction of an agent system architecture. The approach begins with the study of system requirements and the identification of major components of a system in conjunction with the requirements. The components have to be assessed individually and causal links have to be known before the respective components are added to the architecture. This is an incremental approach which encourages the designer to think and evaluate the components carefully, such as the functionalities, the impact and relative effect of each

component's existence to the overall architecture. To complete the study, alternative architectures have to be investigated and compared.

The presentation of the approach led to the design of a generic architecture for an intelligent agent which is illustrated in chapter five. The modelling of the intelligent agent using the layered methodology, have provided the strength to the architecture. Furthermore, the use of multiparadigm approach enabled the system which was built, to behave more intelligently - imitating the human expert's approach in the employment of various strategies when solving problems. It also highlights the advantages of the generic agent architecture.

A variety of multiagent system architectures which utilise agent technology and distributed problem solving technique were presented in chapter six. Even though the different systems architectures were designed using a similar approach, they each differ in a number of ways due to the reorganising and redefinition of the agents' responsibilities in the system. The differences contribute to the advantages and limitations of each system when compared to the others.

The development of the prototypes for each of the system whose architecture was designed in chapter six were presented in chapter seven. Implementations were carried out in different system environments including a knowledge base and an object oriented environment. The systems developed were applied to the area of power system protection which design protection schemes for a power system.

Chapter eight evaluates all the three system architectures, revealing both similarities and differences in each system implemented. The advantages and limitations of each system were probed and studied to enable a better understanding of each system architecture.

Chapter nine presents a discussion on the work that has been completed and gives a review of the thesis. Conclusion based on the complete thesis work is also given. To close the thesis, an outline of recommendations are given as scope for future work.

9.1 Contributions of this Research

There is a growing number of research projects and application systems being developed involving intelligent agents using distributed problem solving technique. These methodologies seem to represent the present and future direction for developing intelligent systems in many areas including medical, manufacturing, diagnosis, networking and engineering. Distributed problem solving technique employs knowledge sources, also known as agents or expert systems, and allow knowledge bases to reside in different environments or platforms. In some systems, multiparadigm techniques are incorporated to make the system more powerful, flexible and robust.

There are not many systems built which utilise layered architectures. However, as communication becomes more entrenched in multiagent systems, layered architecture will begin to play a more eminent role in these systems. Layering is a powerful technique for the design of resource-bounded agents as it

combines a modular structure with a clear control methodology. Moreover, it supports the modelling of different levels of abstraction, reasoning and complexity of knowledge representation. Furthermore, the layering approach offers a more natural, flexible and versatile approach for the modelling of intelligent systems. While the few known systems which employed layered architecture allowed communication between the adjacent layers, the layered architecture in this research has used a selector to conduct the communications between the layers.

Designing and constructing the generic agent architecture has resulted in the development of an adaptive and intelligent agent. The agents developed in the implementation of a multiagent system, possess and demonstrate the natural capability and skills of an expert when making decisions and solving problems.

During problem solving, a number of inference mechanisms may be invoked during the problem solving process. Instead of allowing the layers to take charge of the communication with their adjacent layers, a selector has been introduced which is wholly responsible for the communication between the layers. Its responsibilities include selecting the appropriate inference mechanism and communicating with the appropriate layer by message passing. The purpose of such arrangement is to allow the system to have more control in the management and executions of the multiinference mechanisms in the `Inference Knowledge`.

The implementation of any system could be carried out relatively easily once the system architecture has been designed and verified. If the architecture is well designed, the system implemented would operate in a consistent and

expected manner. Furthermore, the system would be robust, flexible and more adaptive to most changes. On the other hand, if the architecture is ill-designed, any future changes to the system would only bring more problems and other undesirable effects and which may lead to the downfall of the system.

This research has applied the new and popular technologies, such as agent technology, distributed problem solving technique, multiparadigm approach and layering technique to the development of a multiagent system in power system protection. However, in order to accomplish the successful implementation of the system, detailed study and design of architecture was carried out initially.

In this research work, the application domain chosen for the implementation of the intelligent multiagent system is power system protection. This area is selected because power system protection represents a class of varied and interesting engineering designs. The growth of distributed intelligent systems in the electrical and power areas has been slow, but nevertheless, it is growing. As far as literature survey reveals, there is no agent based system developed in the area of power system protection as yet. Most of the developed systems are mainly expert systems and a small number of knowledge based systems. They are known as production rules systems that commonly employ forward or backward chaining process as the inference. However, these conventional systems are becoming more unpopular and undesirable nowadays. The cause of this dissatisfaction is due to a number of factors :

- rapid advancement in today's technology;
- increase in the expectations of a system;
- frequent changes to system requirements;

- escalating demand for more intelligent systems.

9.2 Conclusion

As has already been discussed, the evaluation of an architecture represents a very difficult task. In this investigation, three architectures have been designed, developed and compared. Hence, some form of evaluation of the architectures was made possible. Such evaluation is viewed as important and necessary to substantiate the architecture designed for Intelligent System for Power Protection (ISPP). As stated in chapter three, the design of an architecture does not have much weight without further exploration. Hence, it is important to provide some explanations by showing or indicating how the architecture is better or worse than other designs and to reveal it's trade-offs.

In order to understand and appreciate an architecture fully, it is advocated that the components which made up the system should be explored. Exploring, as explained in chapter three, involves examining and studying each individual component and how it fits into the architecture to meet the system's requirements, to understand its contribution or functionality, and how the overall system performs with and without the existence of the particular component. When the architecture of the system can withstand the scrutiny, then it may be concluded that the design of the architecture is a good one. Most important of all, assuming other things being equal, the system to be built would be efficient, robust, reliable and able to meet all the specified requirements.

The number of research projects in the study and design of architecture in power system protection is very small. Majority of the expert systems or intelligent systems built do not follow any formal approach to the design of the system architecture. In some cases, it is not surprising to find some of the systems do not even have a set of proper specifications to start with. This consequently results in badly designed systems - which tend to break down easily as they can not stand up to rigorous tests or future changes.

In the area of power systems, the development and construction of the majority of the systems do not follow any formal approach to the study and design of the system's architecture. The construction of an intelligent system is no simple task. The development life cycle of a system includes the study of problem domain, analysis requirements, design, implementation, testing and maintenance. While every phase is important to ensure the successful implementation of the system, the design of the architecture should also be emphasised.

Moreover, most of the systems developed in power system area employed traditional techniques and have mainly relied on rules to represent the domain knowledge. Production rule systems have known to be rigid and brittle. Therefore, one of the major concerns of developing systems using the traditional approach is that the knowledge associated with the system is not amenable to automatic adaptation. Furthermore, changes to the system is made more difficult because the architecture of the system has not been properly designed and constructed to allow for future modifications or expansions.

This thesis looks into the aforementioned concerns and problems. The result is the introduction of a novel approach to the study and design of system architecture. This approach is then applied to the development of an intelligent system for power system protection, ISPP, which designs appropriate protection schemes for a power systems or parts of a power system. In addition to using the approach, the ISPP also utilises various strategies and techniques such as agent technology, distributed problem solving technique and multireasoning paradigm reasoning strategy. The employment of multiple artificial intelligence techniques and strategies strengthens the system making it more robust and at the same time, more flexible to automatic adaptations to new contexts and expansions as well.

ISPP has been evaluated and compared to some other alternatives. The design of the agent architecture using the layered methodology and incorporating a multiparadigm approach , has strengthened the development of the agent architecture. Furthermore, the behaviour of the agent is constructed in a way to simulate the behaviour of a human expert and thus, it is able to reason intelligently and follow the natural approaches taken by an expert during problem solving.

The ISPP agents employ various reasoning strategies and switch between them whenever necessary - making the agents more intelligent and enabling them to adopt the human's reasoning capability to assist them in problem solving. This is one of the interesting features of ISPP which allows the agents to be more versatile and adaptable to dynamic changes in the environment.

The system knowledge is represented and maintained in multiple ways within an agent- in the form of experiences in the form of case library (memory) and as facts in the domain knowledge. This is a sophisticated way of representing and managing knowledge base of an intelligent system because such distribution of knowledge enables the system's knowledge to expand or be modified easily without causing any ill effects to the other parts of the system. Furthermore, the system is made more modular and easier to manage.

In addition to the above points and given the discussion and arguments presented, it may be strongly concluded that the generic agent architecture is quite superior to other architectures. It can withstand the scrutiny that was placed on the architecture by the study of the components and the comparisons made with other alternatives. It has also proved to be a robust and efficient system which is amenable to modifications or adaptations. It is also an adaptive architecture which could be easily modified and adapted to suit the development of any system in a number of domains.

The successful implementations of a prototype for each system architecture showed the robustness and flexibility of the systems because of the employment of the adaptive and intelligent agents. However, there also exists limitations as well as advantages of each individual system compared with the others. This clearly indicates that the good architecture design actually contributes to the robustness of a developed system. This indication also goes to prove that the study and design of the architecture is an important key aspect in the development of a system which must not be overlooked.

9.3 Scope for Future Work

The prototype systems implemented shows promising potential for a fully operational system. The systems could include further enhancements such as programming in an object oriented knowledge base environment such as CLOS, provision of an X-Window graphic user interface. A friendly graphical user interface would not only enhance the learning capability of the system but would also promote its use.

Additional specialist agents would be introduced to make the system more complete. In this work, busbar agent has been developed; further agents to be added include those which specialise in transformers, generators, motors and other components of a power system which may require protection.

As the system expands, to cover a wider range of protections for a power system, the agent's domain knowledge and case library (memory) would also be expanded and enhanced to allow the system to operate more efficiently and to grow more effectively and smoothly.

The agents in the system could also be further enhanced to operate not only in a homogeneous environment, but also enable the system to operate in a heterogeneous environment or even on different platforms. This enhancement would involve modifications on the Interface agent in order to expand the agent's capability of handling communications with the external entities residing on other platforms.

APPENDICES

Appendix A

A Common Lisp API

The API is meant to be used in a system of agents operating in the Agent Based Software Interoperation (ABSI) architecture. In the ABSI architecture, there are a collection of agents that communicate with each other in Knowledge Query and Manipulation Language (KQML). The architecture supports automatic interoperation of agents based on their machine readable specifications.

The purpose of the API is to provide a simple interface in common lisp to a writer of an agent in the ABSI architecture. The API is implemented as a subroutine library that is loaded into the working lisp environment.

With the message passing paradigm, the agents communicate with each other by sending packages of information. A ***package*** contains information about the sender agent, receiver agent, communication mode, replies and the content of the package.

The general form of a package is :

```
(package  :content      <expression>
          :sender       <agent-name>
          :receiver     <agent-name>
          :reply-with   <identifier>
```

:in-reply-to <identifier>
:commode <commode>)

The content of a package <expression> is a KIF description of an action, which is also called a *performative*.

The name of an agent <agent-name> is assumed to be unique across the entire system.

The :reply-with field indicates if the sender expects a reply to the package.

The package identifier <identifier> is unique to each package. For example, it could indicates the time in seconds.

The field :commode indicates the type of communication for the package.

The <commode> indicates whether the sender expects a single answer or a stream of answers or none at all to the request (package) sent.

APPENDIX B

EPIC and EPILOG

EPIC is a library of Common Lisp subroutines for use in programs that manipulate information encoded in KIF (Knowledge Interchange Format). It includes translators to convert expressions from one form to another, pattern matchers of various sorts, and subroutines to create and maintain KIF knowledge bases. EPIC does not include any complete inference subroutines for KIF; but it is a good basis for other softwares to be built on it. For example, EPILOG software library is built using EPIC.

EPILOG is a library of Common Lisp subroutines that implement an efficient inference procedure for information encoded in SIF (Simplified Interchange Format). The inference procedure used in EPILOG is based on a reasoning technique called *model elimination.* The procedure closely resembles that of PROLOG; but unlike that of PROLOG, the procedure used in EPILOG is sound and complete for the entire language. That is, all consequences the procedure derives are correct and the procedure can derive all correct consequences of the information it is given.

SIF is a subset language of KIF (i.e., all expressions in SIF are expressions in KIF) but not the reverse. SIF is every bit as expressive as KIF (i.e., for any set of KIF sentences), there is an equivalent set of SIF sentences, which can be derived automatically. These transformations are provided in the EPILOG subroutines.

As subroutine libraries, both EPIC and EPILOG do not run in a standalone fashion. It is designed to be loaded into a running version of Common Lisp using the `load` routine. Once the libraries are loaded, all the variables and subroutines can be used and invoked. In the test implementation of ISPP, both EPIC and EPILOG are being loaded on Lucid Lisp.

Appendix C

Agents' Knowledge Base

The following listing shows the contents of the Bus and Line agent knowledge bases. The 'save' command is used to add facts to a specified 'theory'. A 'theory' is a collection of facts stored in it, and can be viewed as a name for these facts. In this listing, the theories are BusDK and LineDK.

;;;;;;;; ***BUS Agent Knowledge Base*** ;;;;;;;

;;;; ***Unit Protection where backup is not possible/necessary.***

(save '(high-impedance (listof dedicated saturate common)) 'BusDK)

(save '(high-impedance (listof dedicated notsaturate common)) 'BusDK)

(save '(medium-impedance (listof dedicated saturate notcommon)) 'BusDK)

(save '(medium-impedance (listof notdedicated saturate common)) 'BusDK)

(save '(medium-impedance (listof notdedicated saturate notcommon)) 'BusDK)

(save '(low-impedance (listof dedicated notsaturate common)) 'BusDK)

(save '(low-impedance (listof dedicated notsaturate notcommon)) 'BusDK)

(save '(low-impedance (listof notdedicated notsaturate notcommon)) 'BusDK)

(save '(low-impedance (listof notdedicated notsaturate common)) 'BusDK)

;;;; ***Backup Protection For DownStream.***

(save '(idmt (listof current-type nonunit)) 'BusDK)

(save '(idmt (listof impedance-type nonunit)) 'BusDK)

(save '(definite (listof current-type unit)) 'BusDK)

(save '(idmt (listof current-type unit)) 'BusDK)

(save '(definite (listof impedance-type unit)) 'BusDK)

(save '(idmt (listof impedance-type unit)) 'BusDK)

(save '(definite (listof impedance-type nonunit)) 'BusDK)

(save '(definite (listof current-type nonunit)) 'BusDK)

;;;;;;;;; ***LINE Agent Knowledge Base*** ;;;;;;;

;;;; ***Unit Protection where there is communication link between two ends of***

;;;; ***the line.***

(save '(pilot-wire (unit comm-link)) 'LineDK)

(save '(current-diff (unit comm-link)) 'LineDK)

(save '(distance-comm-permissive (unit comm-link)) 'LineDK)

(save '(distance-comm-blocking (unit comm-link)) 'LineDK)

(save '(phase-comparison (unit comm-link)) 'LineDK)

;;;; ***Non-Unit Protection.***

(save '(overcurrent (nonunit comm-link)) 'LineDK)

(save '(overcurrent (nonunit no-link)) 'LineDK)

(save '(dist-impedance (nonunit comm-link)) 'LineDK)

(save '(dist-impedance (nonunit no-link)) 'LineDK)

(save '(dist-definite (nonunit comm-link)) 'LineDK)

(save '(dist-definite (nonunit no-link)) 'LineDK)

(save '(dist-inverse (nonunit comm-link)) 'LineDK)

(save '(dist-inverse (nonunit no-link)) 'LineDK)

Appendix D

Edited Run of the Program

// The program starts with the IA interacting with the user and requesting

// initial information about the power system to be designed.

A SYSTEM FOR POWER PROTECTION

****

Component(s) where new protection schemes need to be designed, starting with the downstream component:

1. BUS.
2. GENERATOR.
3. MOTOR.
4. LINE.
5. TRANSFORMER.
6. CANCEL/EXIT.

// After a brief initial interaction with the user, the IA communicates with the

// Control agent (CA) using the following message package, regarding the new

// designs required.

```
//      (package :content    '(start-design
                                    buscasekb linecasekb coordinatorcasekb)
                 :sender     'interface-agent
                 :receiver   'coordinator)
```

// Using the example as illustrated in Chapter 7, Section 7.3.7 , the user requires a
// coordinated protection scheme for a bus and a line where the line is located
// upstream of the bus. The bus is required to provide backup protection to an
// existing system located downstream from the bus, while the line requires a backup
// protection from an existing system located upstream from it.

Given the above requirements, the CA then decomposes the task and sends requests to the appropriate Design agents eg. Bus/Line agents (BA/LA) to design protection schemes for the bus and line components.

```
//      (package :content    '(design-bus
                                    buscasekb linecasekb coordinatorcasekb)
                 :sender     'coordinator
                 :receiver   'bus
                 :reply-with 'bus-completed)

//      (package :content    '(design-line
                                    buscasekb linecasekb coordinatorcasekb)
                 :sender     'coordinator
                 :receiver   'line
                 :reply-with 'line-completed)
```

// Upon receipt of the above request, the BA/LA will, in turn, send a message
// to the IA requesting for more information regarding the component for
// which protection is to be designed.

// The message takes the form :-

```
//      (package  :content    '(need-information
                                    buscasekb linecasekb coordinatorcasekb)
                  :sender     <agent-name>@
                  :receiver   'interface-agent
                  :reply-with 'proceed)
```

// The :reply-with field indicates that the BA/LA will wait for a reply from the IA

// before proceeding with the design.

// Some of the questions asked by the IA include :-

- Name of <component>:
- Does the <component-name> needs to provide remote back-up protection for downstream component? (y n)
- How many current transformers are to be used for <component-name>?
- Does <component-name> has dedicated cts available? (y n)
- Are the cts of common ratio? (y n)
- Are the cts likely to saturate? (y n)
- Circuit breaker's operating time (ms):
- Total clearance time required (ms):

// When the BA and LA receive a reply from the IA with :in-reply-to field

// containing 'proceed, *they will commence the design process. Each agent will*

// first attempt to retrieve similar designs from its memory and if successful, will

// adapt the retrieved cases to the requirements of the current system.

@ The agent-name represents bus or line.

// Otherwise, the agent will construct the protection scheme from first principles.

*// **When agent completes the requested design, it then sends a reply to the CA with*

// :in-reply-to field containing **'bus-completed** *or* **'line-completed.**

// The following listings show the adapted bus and line cases as well as the

// information regarding applicable relays for the bus protection scheme.

BUS CASE :

((SCHEME (DEFINITE-TIME LOW-IMPEDANCE EARTHPROT))

(BUS BUS-A)

(DOWNPROT Y) (DOWNPROTECTION 11000 CURRENT-TYPED UNIT)

(DOWNPROTDETAILS DEFINITE-TIME 100 200 300)

(UPPROT Y) (CTNUMBER 2)

(CTCHARACTERISTICS DEDICATED NOTSATURATE NOTCOMMON)

(CASEDETAILS 11000 200 1000 1000 100 100 50 120) NIL)

LINE CASE :

((SCHEME (PILOT-WIRE CURRENT-DIFF DISTANCE-COMM-PERMISSIVE

DISTANCE-COMM-BLOCKING PHASE-COMPARISON))

(LINE LINE-AB)

(DOWNPROT Y) NIL NIL

(UPPROT Y) (UPPROTDETAILS 1 100 200 300 500)

(CTNUMBER 1)

(CTCHARACTERISTICS DEDICATED SATURATE NOTCOMMON)

(CASEDETAILS 11000 200 1000 1000 1000 1000 60 120))

** If no similar designs retrieved, the BA will send a request to the IA, requesting for more details regarding the system to be designed. This details would be provided by the user to the IA.
The IA returns a reply to the awaiting BA to enable it to proceed and construct a new design.

Printing Relays List :-

****

LOW-IMPEDANCE :

((LISTOF MBCZ LOW_IMPEDANCE-RELAY GEC))

EARTHPROT :

((LISTOF CMU21 EARTHFAULT-SENSITIVE_POLARIZED_ELEMENT_TYPE GEC)

(LISTOF CTU15B EARTHFAULT-STATIC_RELAY_TYPE GEC)

// ***When the CA receives the reply that the designs have been completed, it will***

// ***first of all, attempt to retrieve similar coordinated cases from its memory.***

// ***Otherwise, the CA will coordinate the previously adapted Bus and Line protection***

// ***schemes.***

// ***In this example, the CA will need to coordinate :-***

// ***(i) The Bus-A with an unknown downstream component and***

// ***the Line-AB located upstream.***

// ***(ii) The Line-AB with an unknown upstream component.***

// ***After going through series of processes which include retrieving, adapting and***

// ***coordinating using first principle basis (if necessary), the results are shown as***

// ***follows:***

Note:

(THE BUS-A DOES NOT COORDINATE WITH ITS DOWNSTREAM PROTECTION.)

(THE TIME SETTING BETWEEN THE BUS-A AND ITS DOWNSTREAM COMPONENT HAS TO BE RESET TO ALLOW THE MARGINAL SAFETY OF 0.4 SEC)

(THE LINE-AB DOES NOT COORDINATE WITH BUS-A. THE SETTING TIME OF EITHER LINE-AB OR/AND BUS-A HAS TO BE RESET TO ALLOW THE MARGINAL SAFETY OF AT LEAST 0.4 SEC.)

(THE LINE-AB DOES NOT COORDINATE WITH THE UPSTREAM COMPONENT THE SENSITIVITY CRITERIA IS NOT SATISFIED.)

"The coordinated cases are : "

Bus Protection Scheme:

((SCHEME (DEFINITE-TIME LOW-IMPEDANCE EARTHPROT))

(BUS BUS-A)

(DOWNPROT Y) (DOWNPROTECTION 11000 CURRENT-TYPED UNIT)

(DOWNPROTDETAILS DEFINITE-TIME 100 200 300)

(UPPROT Y)

(CTNUMBER 2)

(CTCHARACTERISTICS DEDICATED NOTSATURATE NOTCOMMON)

(CASEDETAILS 11000 200 1000 1000 100 100 50 120) NIL)

Line Protection Scheme:

((SCHEME (PILOT-WIRE CURRENT-DIFF DISTANCE-COMM-PERMISSIVE DISTANCE-COMM-BLOCKING PHASE-COMPARISON))

(LINE LINE-AB)

(DOWNPROT Y) NIL NIL

(UPPROT Y) (UPPROTDETAILS 1 100 200 300 500)

(CTNUMBER 1)

(CTCHARACTERISTICS DEDICATED SATURATE NOTCOMMON)

(CASEDETAILS 11000 200 1000 1000 1000 1000 60 120))

"Press any key to continue."

// ***The user can then quit or continue with new designs.***

Appendix E

Blackboard Architecture

The blackboard architecture is based on a metaphor of group problem solving first introduced by Newell and later re-interpreted by Simon [Craig 87a]. The blackboard metaphor expanded upon the metaphor by Newell :

> *" Metaphorically, we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it and to judge when he has something worthwhile to add to it. "*

The blackboard architecture is considered to be a general model for problem solving. According to Craig [Craig 87b], the blackboard has four defining elements :

(i) entries, which are intermediate results generated during problem solving;
(ii) knowledge sources, which are independent, event driven, processes which produce entries;
(iii) the blackboard, which is a structured global database which mediates
(iv) knowledge source interactions and organises entries;
(v) an intelligent mechanism which describes if and when particular knowledge

(vi) sources should generate entries and record them on the blackboard.

Craig [Craig 87b] further summarised the blackboard architecture to the following eleven points :

1. Problem solving activity generates a set of intermediate results which are represented as objects, called entries, with attributes and values.
2. Entries may have user specified relationships with other entries.
3. All entries have the relational attributes: abstract/refines and adjacent-to. These attributes define the vertical and horizontal structure of the blackboard.
4. All entries are recorded in a global database called the blackboard.
5. The blackboard structure includes partitions for different levels of abstraction and solution intervals.
6. The blackboard may have additional, user defined structure.
7. Independent knowledge representing processes, called knowledge sources, generate, modify and record entries on the blackboard.
8. Each knowledge source has a condition and an action. The condition matches a hypothetical configuration of entries on the blackboard, performs computation and is a predicate. The action performs computation and generates blackboard modifications.
9. Only triggered knowledge sources can be executed.
10. An intelligent scheduler determines which triggered knowledge sources should execute their actions.
11. The scheduler can base its decisions on user determined criteria such as the characteristics of the triggered knowledge sources, the

utility of the proposed action, information about the general blackboard state, characteristics of the problem or information about previous control decisions.

APPENDIX F

Case Based Reasoning

Case based reasoning (CBR) in AI has its roots in the works of Roger Schank on dynamic memory. It plays a central role in that a reminding of earlier situations (episodes or cases) and situation patterns (scripts, memory organisation packets MOPs) have in problem solving and learning [Aamodt and Plaza 94]. According to Pu [Pu 93], CBR is an experience-based method. As Schank put it :

> *"... an expert is someone who gets reminded of just the right prior experience to help him in processing his current experiences. "*

The cases which correspond to the past problem solving experiences, make up the knowledge source or the knowledge base of a system. And hence, such systems are called case based reasoning systems.

Case based reasoning is a problem solving paradigm that is fundamentally different from other major AI approaches in many respects [Aamodt and Plaza 94]. It is a technique that builds a case library consisting of previous experiences stored as cases. Problem solving is accomplished by retrieving similar cases and adapting the solutions to the problem at hand, and finally evaluating the proposed solution. While many other paradigms rely solely on

general knowledge of a problem domain, or making associations along generalised relationship between problem descriptors and conclusions (e.g., rule-based reasoning, model based reasoning), CBR utilises or employs the specific knowledge of previous experienced, concrete problem situations (cases) [Aadmodt and Plaza 94]. Another important feature that differentiates CBR from other problem solving paradigms is its incremental and sustained learning approach. Every new experiences is retained in its case library, also known as case memory, making it immediately available for future problem solving.

CBR paradigm has grown to be one of the most popular widespread interest in the research world besides agents and distributed artificial intelligence. It becomes a favourable approach because CBR resembles very closely to the natural way of reasoning that people employ in their daily decision making process. This is true as in practical situation, problem solving seldom or very rarely begins from first principle. People always refer to previous experiences before they start working on any new problem.

Basically, the fundamental or central tasks of a CBR system are :

- Identify the current problem situation;
- Retrieve similar cases from the case library;
- Adapt retrieved solutions to the current problem;
- Evaluate the proposed solution;
- Update system and learn from the new experience.

The areas of applications have grown and the systems that have been developed include those which use cases to resolve disputes, design, planning, legal

reasoner, chess game, investments, predictions and diagnosis [Dutta and Bonissone 93, Carville *et al.* 93, Lewis 93, Maher and Zhang 93, Rissland *et al.* 94]. The reasoning process commonly involves four issues [Mott 93]:

(i) case representation - concerns with the structure used to represent a case, the features of a case and how they can be expressed (numbers, booleans, date, etc.).

(ii) case indexing - refers to the indexing schemes used in order to retrieve the most relevant or similar cases.

(iii) case retrieval - concerns with the organisation of the cases for efficient retrieval and the retrieval methods used.

(iv) case adaptation - refers to the way retrieved cases are modified or adapted, when the adapted cases should be stored and how the knowledge for adaptation is obtained.

Cases can be represented as rules, frames, slots or embedded objects. The cases may be stored as a complete unit or decomposed into sub-units and distributed within the knowledge structure. This decision has a direct effect on the degree of difficulty in retrieving and adapting cases [Pu 93]. Storing the cases in decomposed sub-units has its benefits over storing them as a complete unit. Firstly, a large case broken down into pieces and kept separately enables the pieces to be used to address different aspects of a new problem. Secondly, problem solving becomes more efficient by using and combining the different pieces of different cases together to form the new solution without affecting any previous cases.

The choice of indexing, retrieval algorithm and memory organisation of the case library can help to maintain or improve the system performance. But there has always been an awareness of the indexing and retrieval problems especially in applications where the case library or the case memory can grow to be a very large library containing hundreds or even thousands of cases. However, the issue of *search space* is seldom or rarely looked into or discussed. The *search space* is defined as the size of the base containing the objects to be searched. Essentially, most of the case based systems start their retrievals from the top of the case library and narrowing their search path as they proceed with the retrieval process. One of the main focus of this thesis is the issue of reducing the search space even before retrievals begin and proposes a method to accomplish that in order to improve the efficiency of retrievals.

Case adapting is part of case reusing [Aamodt and Plaza 94]. This process involves reusing previous case solution (transformational reuse) and previous method to construct the solution (derivational reuse). In other words, part of the previous case solution is transferred and modified to solve the current problem.

As stated by Decker [Decker 87], one of the most powerful aspects of AI approach to problem solving is the ability to deal with uncertain and incomplete information. Therefore, the approach outlined here proposes the integration of DPS and case-based reasoning (CBR) techniques. One of the many advantages of CBR systems is that it can handle incomplete information or missing data quite well [Stottler 94]; that is, the features of the missing values will not be used in the retrieval process. The retrieved cases which possess a variety of values for the missing feature(s) can determine its influence and importance on the solution proposed. If the solution or outcome of the retrieved case(s) is

acceptable, then the missing value has proven to be unimportant. Otherwise, a new solution can be derived using some other reasoning technique.

REFERENCES

[AAAI 94] Proceedings of the Twelfth National Conference on Artificial Intelligence, Volume One and Two, Washington, July 1994, American Association for Artificial Intelligence.

[AIC 93] Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science, September 1993, Queen's University, Belfast.

[APSCOM 91] Proceedings of the International Conference on Advances in Power System Control, Operation and Management, 1991, Hong Kong.

[AUPEC 94] Proceedings of the Australasian Universities Power Engineering Conference, Adelaide, Australia, September 1994, University of South Australia.

[Aamodt and Plaza 94] Aamodt A., Plaza E.,
Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *AICOM* vol.7 (1), March 1994, pp.39-59.

[Adler *et al.* 92] Adler M., Durfee E., Huhns M., Punch W., Simoudis E.,
AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents, *Proceedings of AAAI 1992*, pp.39-40.

[Atkinson *et al.* 93] Atkinson M.P., Sjoberg D.I.K., Morrison R.,
Managing Change in Persistent Object Systems, *Lecture Notes in Computer Science 1993*, pp.315-338.

[Austin 62] Austin J.L.
How to do things with words. Harvard University Press, Cambridge, MA, 1962.

[Bardasz and Zeid 93] Bardasz T and Zeid I.,
DEJAVU: Case-Based Reasoning for Mechanical Design, *AI EDAM* 7(2), 1993, Academic Press Limited.

[Beaudoin *et al.* 93] Beaudoin L.P., Paterson C., Shing E., Sloman A.and Wright I.,
A Summary of the Attention and Affect Project, *Technical Report 1993*, Cognitive Science Research Centre, University of Birmingham.

[Bond and Gasser 88] Bond A.H. and Gasser L.,
An analysis of problems and research in DAI. *Readings in Distributed Artificial Intelligence*. San Mateo, CA, Morgan Kaufmann Publishers, 1988, pp.3-35.

[Brown *et al.* 96] Brown R.E., Gupta S., Christie R.D. and Venkata S.S.,
A Genetic Algorithm for Reliable Distribution System Design, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.29-34.

[Burstein *et al.* 94] Burstein F.V., Smith H.G., Arnott D.R. and O'Donnell P.A.,
On the Nature of Intelligent Decision Support Systems, *Technical Report 1994*, Monash University, Australia.

[Bussmann and Muller 93] Bussmann S. and Muller J., A Communication Architecture for Cooperating Agents. *Computers and Artificial Intelligence*, 12(1), 1993, 37-53.

[Buttler 95] Buttler J., Maintaining Quality and Quantity Production Constraints in a Multi-Agent Batch Scheduling Environment, *Proceedings of IEA95AIE*, Melbourne 6-8 June 1995, pp.679-685.

[Carville *et al.* 93] Carville F., Dubitzky W. and Hughes J., Case-Level Knowledge Modelling in CBR, *Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science*, September 1993, Queen's University, Belfast, pp.217-227.

[Case-Based Reasoning 91] Proceedings of Case-Based Reasoning Workshop, Washington, May 1991, DARPA/ISTO.

[Celko 93] Celko J., Genetic Algorithms and Database Indexing, *Dr. Dobb's Journal*, Apr 1993, vol.15, no.4, pp.30-34.

[Chandra 92] Chandra D.N., Innovative Design Systems, Where are we and where do we go from here? Part 1: Design by association, *The Knowledge Engineering Review*, vol.7:3, 1992, pp.183-213.

[Chang *et al.* 96] Chang C.S., Chen J.M. and Liew A.C., Power System Fault Diagnosis using Fuzzy Sets for Uncertainties Processing, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.333-338.

[Chen *et al.* 94] Chen J., Wang P., Fu S., Wang M. and Yu E., Development of an Object-Oriented Expert System for Power System On-Line Restoration after a Blackout, *ICPST 1994*, pp.1061-1065

[Choi *et al.* 95] Choi D.S., Kim C.S. and Hasegawa J., An Application of Genetic Algorithms to the Network Reconfiguration in Distribution for Loss Minimisation and Load Balancing Problems, *Proceedings of International Conference on Energy Management and Power Delivery 1995*, EMPD 1995, Singapore, pp.376-381.

[Cockburn and Jennings 94] Cockburn D., Jennings N.R., ARCHON: A Distributed Artificial Intelligence System for Industrial Applications, *Technical Report 1994*, University of London.

[Craig 87a] Craig I., The Blackboard Systems, *Research Report 110*, Department of Computer Science, University of Warwick, United Kingdom, 1987.

[Craig 87b] Craig I., The Blackboard Architecture: A Definition and Its Implications, *Research Report 87*, Department of Computer Science, University of Warwick, United Kingdom,1987.

[da Silva and Zebulum 96] da Silva A.L. and Zebulum R.S., An Integration of Neural Networks and Fuzzy Logic for Power Systems Diagnosis, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.237-241.

[Dash and Panda 95] Dash P.K. and Panda S.K., Fuzzy Tuning of DC Link Controllers, *Proceedings of International Conference on Energy Management and Power Delivery 1995*, EMPD 1995, Singapore, pp.370-375.

[Decker 87] Decker K.S, Distributed Problem-Solving Techniques: A Survey, *IEEE Transactions of Systems, Man and Cybernectics* vol. SMG- 17(5), Sep/Oct 1987, pp.729-740.

[Devapriya *et al.* 92] Devapriya D.S., Descotes-Genon B., Ladet P., Distributed intelligence systems for FMS control using objects modelled with petri nets (scope blackboard), *Manufacturing Systems* 1992, pp.73-77.

[Durfee and Rosenschein 94] Durfee E.H. and Rosenschein J., Distributed Problem Solving and Multiagent Systems: Comparisons and Examples, *Proc. of the 13th International Workshop on DAI*, Lake Quinalt, WA, 1994, 94-104.

[Durfee *et al.* 85] Durfee E.H., Lesser V.R. and Corkill D.D., Coherent Cooperation Among Communicating Problem Solvers, *Technical Report 85-15*, Department of Computer and Information Science, University of Massachusetts, Massachusetts, April 1985.

[Dutta and Bonissone 93] Dutta S., Bonissone P.P., Integrating Case- and Rule-Based Reasoning, *International Journal of Approximate Reasoning* (8) 1993, pp.163-203.

[Eickhoff *et al.* 91] Eickhoff F., Handschin E. and Hoffmann W., Knowledge Based Alarm Handling and Fault Location in Distribution Networks, *1991 Power Industry Computer Application Conference*, 17th PICA Conference, Baltimore, USA, pp.358-364.

[Elders 96] Elders I.M., The Application of an Intelligent System to Power Network Operation and Control, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.170-174.

[El-Sharkawi and Huang 96] El-Sharkawi M.A. and Juang S.J., Development of Genetic Algorithm Embedded Kohonen Neural Network for Dynamic Security Assessment, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.44-49.

[EMPD 95] Proceedings of the International Conference on Energy Management and Power Delivery 1995, Proceedings of EMPD 1995, Singapore,

[Enns *et al.* 92] Enns M.K., McGuire P.F., Ramaswami R., Arbor A., CAPE: The Computer-Aided Protection Engineering System, *Proceedings of the American Power Conference*, 13-15 Mar 1992, vol.2, pp.1084-1089.

[Epilog 94] *EPILOG 1.0 for LISP*, Draft Report, Epistemics 1994.

[ESAP 93] Expert System Application to Power Systems IV, Proceedings of ESAP 1993, Melbourne, Australia.

[EWCBR 93] First European Workshop on Case-Based Reasoning, November 1993, University of Kaiserslautern, Germany.

[Finin *et al.* 94] Finin T., Fritzson R., McKay D. and McEntire R.,
KQML as an Agent Communication Language. *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94)*, ACM Press, November 1994. (To appear.)

[Finin *et al.* 93] Finin T., Weber J., Wiederhold G., Genesereth M., Fritzson R., McKay D., McGuire J., Pelavin P., Shapiro S. and Beck C.,
Specification of KQML Agent-Communication Language. Enterprise Integration Technologies, Palo Alto, CA, *Technical Report EIT TR 92-04*, updated July 1993.

[Gaiti and Pujolle 92] Gaiti D., Pujolle G.,
An Intelligent IN, *International Journal of Network Management*, December 1992, pp.183-189.

[Galletly 92] Galletly J.E.,
An Overview of Genetic Algorithms, *Kybernetes*, vol.21 no.6, 1992, pp.26-30.

[Gasser and Huhn 87] Gasser G. and Huhn M.N., *Distributed Artificial Intelligence*, Vol. II. Research Notes in Artificial Intelligence, San Mateo, CA, Morgan Kaufmann Publishers, 1987.

[GEC Alsthom 87] GEC Alsthom, *Protective Relays Application Guide*, GEC Alsthom Measurements Limited, 1987.

[Genesereth and Fikes 92] Genesereth M.R. and Fikes R.E. Knowledge Interchange Format, version 3.0, *Reference Manual*, Logic Group, Logic Report-92-1, Stanford University, 1992.

[Genesereth and Ketchpel 94] Genesereth M.R. and Ketchpel S.P., Software Agents, *Communications of the ACM*, July 1994 vol.37(7), pp.48-53.

[Goldberg 89] Goldberg D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co. Inc., 1989.

[Goodwin 93] Goodwin R., Formalizing Properties of Agents, *Technical Report No.CMU-CS-93-159, May 1993*, School of Computer Science, Carnegie Mellon University.

[Gora and Kacejko 89] Gora S., Kacejko P., Application of expert system for determination of protective devices' settings, *Proceedings 2nd Symposium Expert System Applications to Power Systems (ESAPS)*, Seattle, WA., USA, 1989, Electric Power Research Institute, Palo Alto, CA., pp 251-256.

[Grice 89] Grice P.,
Studies in the Ways of Words. Harvard University Press, Cambridge, MA, 1989.

[Gunther and Lamberts 94] Gunther O. and Lamberts J.,
Object Oriented Techniques for the Management of Geographic and Environmental Data, *The Computer Journal*, vol.37 no.1, 1994, pp.16-25.

[Hagg and Ygge 95] Hagg S. and Ygge F.,
Agent Oriented Programming in Power Distribution Automation, *Licentiate Thesis 95*, Department of Computer Science, Lund University, Sweden, 1995.

[Hair *et al.* 92] Hair D.C., Pickslay K., Chow S., Explanation-Based Decision Support In Real Time Situations, *Proceedings of the 1992 IEEE International Conference on Tools with AI,* Arlingtion, VA, Nov 1992, pp.22-25.

[Hariri and Malik 96] Hariri A. and Malik O.P.,
Self-Learning Adaptive Network Based Fuzzy Logic Power System Stabilizer, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.299-303.

[Hatta *et al.* 88] Hatta M., Satoh H., Seriwaza Y., Application of Expert System to the Determination of the Operational Coordination of Switch Gears and Power Fuses for Short Circuit Protection, *Proceedings First Symposium Expert System Applications to Power Systems (ESAPS)*, Research Institute Technology, Stockholm, Sweden, 1988, pp.6.27-6.34

[Hayes-Roth 94] Hayes-Roth B., An Architecture for Adaptive Intelligent Systems, *Technical Report 1994*, Stanford University, CA.

[Hayes-Roth *et al.* 94] Hayes-Roth B., Pfleger K., Lalanda P., Morignot P. and Balabanovic M., A Domain-Specific Software Architecture for Adaptive Intelligent Systems, *Technical Report 1994*, Stanford, CA.

[Hill and Schmalenbach 95] Hill S. and Schmalenbach J., An On-Line Design System for Fuzzy Controllers, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, IEA/AIE95, Melbourne, Australia, pp.411-418, 1995.

[Hinterding and Juliff 93] Hinterding R. and Juliff K., A Genetic Algorithm for Stock Cutting: An Exploration of Mapping Schemes, *Technical Report August 1993*, Victoria University of Technology, Australia.

[Hoffmann 94] Hoffmann M.,
Innovations That Migrate: Fuzzy Logic, Why do U.S. innovations migrate overseas for development and application? Copyright 1994 by *SRI International Business Intelligence Program.*

[Holen 96] Holen A.T.,
Future Need for Intelligent Systems in the Changing Utility Environment, *International Conference on Intelligent Systems Applications to Power Systems 1996*, ISAP 1996 Panel Session, Florida, USA.

[Holland 92] Holland J.H.,
Genetic Algorithms, *Scientific American*, July 1992, pp.44-50.

[Huang *et al.* 94] Huang J., Jennings N.R. and Fox J.,
An Agent Architecture for Distributed Medical Care, *Technical Report 1994*, University of Central Lancashire, U.K.

[Huang and Brandon 93] Huang G.Q., Brandon J.A.,
Agents for Cooperating Expert Systems In Concurrent Engineering Design, *AI EDAM* (1993) 7 (3), pp.145-158.

[Huang and El-Sharkawi 96] Huang T.C. and El-Sharkawi M.A.,
Induction Motor Efficiency Maximizer Using Multi-Layer Fuzzy Control, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.109-113.

[Huber and Wu 91] Andreas H., Wu F., Distributed Computing for an Advanced Energy Management System, *Advances in Power System Control, Operation & Management 91*, The Institution of Electrical Engineers, pp.605-611.

[Huhn 87] Huhn M.N., *Distributed Artificial Intelligence*. San Mateo, CA, Pitman/Morgan Kaufmann, 1987.

[IEA/AIE 95] Proceedings of the Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Proceedings of IEA/AIE 95, Melbourne, Australia, June 6-8, 1995.

[ISAP 96] Proceedings of the International Conference on Intelligent Systems Applications to Power Systems, Proceedings of ISAP 96, Florida, USA, Jan. 28 - Feb. 2, 1996.

[Isbister and Layton 94] Isbister K. and Layton T.L., Agents in Review: Examples, Dimensions, and Issues, *Technical Report 1994*, Stanford University.

[Isomursu *et al.* 95] Isomursu P., Rauma T. and Kemppainen S., Applying a Model of Fuzzy Control Software Development Process in Practice, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, IEA/AIE95, Melbourne, Australia, pp.389-390, 1995.

[Jenkins 92] Jenkins W.M.,
The Genetic Algorithm - or can we improve design by breeding?, *Colloquium on Artificial Intelligence in Civil Engineering*, January 1992, Institute of Electrical Engineers, London.

[Jennings *et al.* 93] Jennings N.R., Varga L.Z., Aarnts R.P., Fuchs J., Skarek P.,
Transforming Standalone Expert Systems Into A Community of Cooperating Agents, *Engineering Application in Artificial Intelligence*, 6(4), 1993.

[Jennings 94] Jennings N.R.,
The ARCHON System and its Applications. *Second International Conference on Cooperating Knowledge Based Systems (CKBS-94)* (Invited Paper), Keele, UK, 1994.

[Jiang and Teo 95] Jiang W. and Teo C.Y.,
A Knowledge Based Approach for Bulk Power System Restoration, *Proceedings of International Conference on Energy Management and Power Delivery 1995*, EMPD 1995, Singapore, pp.108-101.

[Kagan and Oliviera 96] Kagan N. and Oliviera C.C.B.,
A Fuzzy Constrained Decision Planning Tool to Model Uncertainties in Multiobjective Configuration Problems, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.271-276.

[Kalam *et al.* 91] Kalam A., Klebanowski A., Poloni D., Knowledge Based System for Transformer Protection, *Proceedings on IEE International Conference on Advances in Power System Control, Operation and Management 1991*, pp.443-448.

[Kalam and Negnevitsky 93] Kalam A., Negnevitsky M., Knowledge Based Approach To Clearing Overloads On The Power System Plant, *Proceedings of Second International Conference on Modelling and Simulation*, Victoria University of Technology, Melbourne, Australia, 12-14 July 1993, pp.355-363.

[Kawahara *et al.* 93] Kawahara K., Sasaki H., Kubokawa J., Sugihara H., Kitagawa M., An Expert System for Supporting Protective Relay Setting for Transmission Lines, *Fifth International Conference on DPSP 1993*, IEE Publications, pp.203-206.

[Kawamura and Wakizono 96] Kawamura T. and Wakizono R., Evaluation of Object Oriented Database for Distribution Network Monitoring System, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.156-161.

[Khedro *et al.* 93] Khedro T, Genesereth M.R, Teicholz P.M, Agent-based framework for integrated facility engineering, *Engineering with Computers* vol.9 1993, pp.94-107.

[Kezunovic *et al.* 91] Kezunovic M., Watson K., Russell B.D., Heller P., Aucoin M.,
Expert System Applications to Protection, Substation Control and Related Monitoring Functions, *Electric Power Research*, 21, 1991, pp.71-86.

[Kolodner 91] Kolodner J.L.,
Improving Human Decision Making through Case-Based Decision Aiding, *AI Magazine*, Summer 1991, pp.52-68.

[Kolodner 93] Kolodner J.L.,
Cased Based Reasoning, Morgan Kaufmann Publishers, Inc., 1993.

[Koton 88] Koton P.,
Reasoning about Evidence in Causal Explanations, *Proceedings Case-Based Reasoning Workshop*, Florida, May 10-13, 1988, Morgan Kaufmann Publishers.

[Lai 89] Lai L.L.,
Development of an Expert System for Power System Protection Coordination, *Fourth International Conference in Power System Protection*, 11-13 April 1989, pp. 310-314.

[Lai and Ma 96] Lai L.L. and Ma J.T.,
Power Flow Control with UPFC using Genetic Algorithm, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.373-379.

[Lashkari *et al.* 94] Lashkari Y, Metral M, Maes P, Collaborative Interface Agents, *Twelve National Conference on AI 1994*, pp.444-449.

[Lee *et al.* 89a] Lee S.J., Yoon S.H., Moon M.C. and Yang J.K., An Expert System for Protective Relay setting of transmission systems, *Proceedings PICA Conference*, Seattle, WA., USA, 1989, IEEE, New York, pp. 296-302.

[Lee *et al.* 89b] Lee S.J., Yoon S.H., Moon M.C. and Yang J.K., PROSET: An Expert System for Protective Relay Setting, *Proceedings IFAC Symposium Power Systems and Power Control Plant*, Korea 1989, pp.1003-1007.

[Lee *et al.* 94] Lee H.J., Park Y.M., Hyun S.H., Chu J.B. and Yoon Y.B., Development of an Intelligent Support System at Local Power Control Center in Korea, *ICPST 1994*, Beijing, China, pp.494-497.

[Lee *et al.* 96] Lee H.J., Lee S.J., Park Y.M., Park J.K. and Choo J.B., An Intelligent Support System for Local Control Center Using Meta-Inference and Reconstruction of Knowledge Base, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.247-252.

[Leung 91] Leung K.W., Computer-Aided Setting Calculation for Distance Zone 2 and Zone 3 Protection, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.152-157

[Lewis 93] Lewis L.,
A Case-Based Reasoning Approach to the Management of Faults in Communications Networks, *1993 IEEE Conference on AI for Applications*, 1993.

[Li *et al.* 91] Li K.K, Cheung C., Xia Y.Q.,
High Speed Digital Distance Protection -Real Time Simulation and Hardware Development, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.95-100.

[Li *et al.* 95] Li T., Guo D., Jayakumar R., Tang Y.Y. and Klasa S.,
A Fuzzy Learning Expert System and Its Applications, *Proceedings of the 6th Australian Joint Conference on Artificial Intelligence*, AI 1995, Melbourne, Australia, pp.119-124.

[Liebowitz 89] J. Liebowitz,
Structuring Expert Systems - Domain, Design and Development, Chapter One, Yourdon Press 1989.

[Liu *et al.* 90] Liu C.C, Ma T.K., Liou K.L. and Tsai M.S.,
Practical Use of Expert Systems in Power Systems, *Expert System Applications to Power System IV*, Melbourne, Australia, January 1990, Plenary Paper.

[Liu *et al.* 94] Liu L., Gao Z., Yang Q., Tong W.,
Construction of Expert System for Designing Protection of Power Transformer, *ICPST 1994*, pp.1380-1384.

[Lo and Nashid 93] Lo K.L. and Nashid I., Expert Systems and Their Application to Power Systems, Part 1: Components and methods of knowledge representation, *Power Engineering Journal*, vol.7 no.1 February 1993, IEE Publication.

[Lu *et al.* 88] Lu C.N., Liu K.C. and Unum M.R., A Knowledge Based Consultation Facility for Power System Application Software, *Ninth CEPSI 1988*, pp.359-368.

[MS 93] Second International Conference on Modelling and Simulation, Melbourne, 1993, Victoria University of Technology.

[McArthur *et al.* 96] McArthur S.D.J., Bell S.C., McDonald J.R., Mather R. and Burt S.M., Knowledge and Model Based Decision Support for Power System Protection Engineers, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.215-219.

[Mac Intosh *et al.* 91] Mac Intosh D.J., Conry S.E., Meyer R.A., Distributed automated reasoning: Issues in coordination, cooperation and performance, *IEEE Transactions on Systems, Man and Cybernetics* vol.21(6) Nov/Dec 1991, pp.1307-1316.

[Maes 94] Maes P, Agents that Reduce Work and Information Overload, *Communications Of The ACM* vol.37 (7) 1994, pp.31-40.

[Maher and Zhang 93] Maher M.L. and Zhang D.M., CADSYN: A Case-Based Design Process Model, *AI EDAM* 7(2), 1993, Academic Press Limited.

[Marin and Banerjee 94] Marin M.A., Banerjee S.N., An Approach to a Knowledge-Based Diagnostic System for Power Equipment Maintenance, *ICPST 1994*, pp.717-720.

[Minton *et al.* 90] Minton S., Carbonell J.G., Knoblock C.A., Kuokka D.R., Etzioni O. and Gil Y., Explanation-Based Learning: A Problem Solving Perspective, *Machine Learning*, Publisher MIT/Elsevier 1990, pp.63-118.

[Mookerjee and Chaturvedi 93] Mookerjee V.S. and Chaturvedi A.R., A Blackboard Control Architecture for Model Selection and Sequencing, *European Journal of Information Systems*, vol.2 no.1, Jan 1993, pp.3-14.

[Mott 93] Mott S., Case-Based Reasoning: Market, Applications, and Fit with other Technologies, *Expert Systems With Applications*, vol.6, 1993, pp.97-104.

[Ngan *et al.* 91] Ngan H.W., David A.K., Lo K.L., Modelling of A Distributed Computer Controlled Multiterminal HVDC System for Dynamic Simulation, *Advances in Power System Control, Operation & Management 1991*, The Institution of Electrical Engineers, pp.599-605.

[Nii 86] Nii H.P., Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of BlackBoard Architectures, The AI Magazine, Summer 1986, pp.38-53.

[Nishiyama *et al.* 92] Nishiyama T, Katai O, Sawaragi T, Iwai S, Horiuchi T, A framework for multiagent planning and a method of representing its plan integration, *Transputer/Occam Japan* 4 1992, pp.161-175.

[O_2System 93] O_2System - A Technical Overview, September 1993.

[Oates *et al.* 94] Oates T., Nagendra Prasad M.V., Lesser V.R., Cooperative Information Gathering: A Distributed Problem Solving Approach, *Technical Report 94-66-version 2*, University of Massachusetts, USA, 1994.

[Oyama 96] Oyama T., Restorative Planning of Power Systems Using Genetic Algorithm with Branch Exchange Method, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.175-179.

[Park *et al.* 96] Park Y.M., Moon U.C. and Lee K.Y., A Power System Stabilization with a Self-Organizing Fuzzy Logic Controller, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.114-118.

[Parlos *et al.* 96] Parlos A., Oufi E., Muthusami J., Patton A. and Atiya A.F.,
Development of an Intelligent Long-Term Electric Load Forecasting System, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.288-292.

[Patil *et al.* 92] Patil R., Fikes R., Patel-Schneider P., McKay D., Finin T., Gruber T. and Neches R.,
The DARPA Knowledge Sharing Effort: Progress report. In B.Nebel, C.Rich and W.Swartout, (eds), Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference (KR '92), San Mateo, CA, Morgan Kaufmann Publishers, November 1992.

[Patterson 90] Patterson D.W.,
Introduction to Artificial Intelligence and Expert Systems, Prentice Hall, 1990.

[Peligry 94] Peligry Y.P.,
An Illustration of the SHADE Concept: The Unit and Dimension Agent, *Technical Report No.KSL-94-24*, Knowledge Systems Laboratory, Stanford University, 1994.

[Pu 93] Pu P.,
Introduction: Issues in Case-Based Design Systems, *AI EDAM* vol.7 (2) 1993, pp.79-85.

[Ravindranath and Chander 77] Ravindranath B., and Chander M.,
Power System Protection and Switchgear, Wiley Eastern Limited, India, 1977.

[Read 93] Read T.,
Systemic Design: A Methodology for Investigating Emotional Phenomena, *Technical Report*, University of Birmingham, UK, 1993.

[Read 94] Read T.,
Applying Systemic Design to the Study of 'Emotion', to be presented at the *AICS'94* conference in Dublin, Ireland, 1994.

[Read and Sloman 93] Read T. and Sloman A.,
The Terminology Pitfalls of Studying Emotion, *Technical Report*, University of Birmingham, UK, 1993.

[Reddy and O'Hare 91] Reddy M., O'Hare G.M.P.,
The Blackboard Model: A Survey of its Application, *Artificial Intelligence Review* (1991) 5, pp.169-186.

[Riecken 94] Riecken D.,
M: An Architecture of Integrated Agents, *Communications of the ACM*, July 1994 vol.37 no.7, pp.107-116.

[Rissland *et al.* 94] Rissland E.L., Skalak D.B. and Friedman M.T.,
Heuristic Harvesting of Information for Case-Based Argument, *Twelve National Conference on AI 1994*, pp.36-43.

[Rossomando 92] Rossomando P.J.,
The Achievement of Spacecraft Autonomy Through the Thematic Application of Multiple Cooperating Intelligent Agents, *Telematics and Informatics*, vol.8 no.3/4 1992, pp.205-219.

[Russell and Watson 87] Russell B.D. and Watson K., Power Substation Automation Using A Knowledge Based System - Justification and Preliminary Field Experiments, *IEEE Transactions on Power Delivery*, vol.PWRD-2 no.4, October 1987.

[Sakata and Iwamoto 96] Sakata M. and Iwamoto S., Genetic Algorithm Based Real-Time Rating for Short-Time Thermal Capacity of Duct Installed Power Cables, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.85-90.

[Searle 69] Searle J.R., *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK, 1969.

[Singh 93] Singh A., Common Lisp API and Facilitator for ABSI, version 2.0.3. Logic Group, *Logic Report-93-4*, Stanford University, 1993.

[Shaw and Fox 93] Shaw M.J. and Fox M.S., Distributed Artificial Intelligence for Group Decision Support, Decision Support Systems (1993), pp.349-367.

[Slade 91] Slade S., Case-Based Reasoning: A Research Paradigm, *AI Magazine*, Spring 1991.

[Sloman 92] Sloman A.,
The Mind as a Control System, *Proceedings of 1992 Royal Institute of Philosophy Conference 'Philosophy and the Cognitive Sciences'*, Cambridge University Press.

[Sloman *et al.* 94] Sloman A. and the Cognition and Affect Group,
Explorations in Design Space, in Proc ECAI94, *11th European Conference on Artificial Intelligence*, Edited by A.G.Cohn, John Wiley, pp 578-582, 1994.

[Sloman 95] Sloman A.,
Exploring Design Space and Niche Space, *International Proceedings 5th Scandinavian Conference on AI*, Trondheim May 1995.

[Smith 85] Smith R.G.,
Report on the 1984 Distributed Artificial Intelligence Workshop, The AI Magazine, Fall 1985, pp.234-243.

[Smith and Davis 81] Smith R.G. and Davis R.,
Frameworks for Cooperation in Distributed Problem Solving, *IEEE Transactions on Systems, Man and Cybernetics*, vol.smc-11 no.1, January 1981, pp.61-70.

[Smith and Poulter 93] Smith H.N. and Poulter K.J.,
The elements of an open KBS infrastructure. *Future Generation Computer Systems* 9, 1993, 349-369.

[Smith and Slade 92] Smith R. and Slade A.,
The Concensus Approach to Intelligent Systems Engineering, *First International Conference on Intelligent System Engineering*, August 1992, pp.86-91.

[Sriyananda and Silva 91] Sriyananda H., Silva A.R.D.S.,
The Use of Distributed Intelligence in the Control of Power Systems Under Emergency Conditions, *Advances in Power System Control, Operation & Management 1991*, The Institution of Electrical Engineers, pp.611-615.

[Stoica and Wingate 93] Stoica A. and Wingate M.,
Fuzzy Modelling and Simulation for a Fencing Robot, Second International Conference on Modelling and Simulation, MS 1993, Melbourne, Australia, pp.199-209.

[Stottler 94] Stottler R.H.,
CBR for Cost and Sales Prediction, *AI Expert* August 1994, pp.25-32.

[Sumic and Vidyanand 96] Sumic Z. and Vidyanand R.,
Fuzzy Set Theory Based Outage Determination, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.204-208.

[Sumic *et al.* 90] Sumic Z., Pistorese T., Venkata S.S., Wei X., Yeh E.C. and Atteri R.,
Implementation of an AI Based Design Tool in a Facility Management System, *Expert System Applications in Power System IV*, Melbourne, Australia, January 1990, pp.428-433.

[Sycara 88] Sycara K.,
Using Case Based Reasoning for Plan Adaptation and Repair, *Workshop of Case Based Reasoning 1988*, Florida, pp. 425-434.

[Sycara 94] Sycara K.,
The Present and Future of DAI: A Study in Enterprise Integration, *Seventh Australian Joint Conference on Artificial Intelligence*, *Artificial Intelligence AI 1994*, Armidale, pp.1-12.

[Tang *et al.* 96] Tang S.K., Dillon T.S. and Khosla R.,
Application of an Integrated Fuzzy, Knowledge Based, Connectionistic Architecture for Fault Diagnosis in Power Systems, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.188-193.

[Tangen and Stoa 96] Tangen G. and Stoa P.,
Decision Support for Hydropower Plant Upgrading: Integrating Multi Attribute Decision Making and Knowledge Based Systems. *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.145-150.

[Taylor *et al.* 90] Taylor A., Hawken A. and Laughton M.A.,
The Use of Knowledge Based Systems in Delivering Real-time Plant Protection and Safety, *Expert System Applications to Power Systems IV*, Melbourne, Australia, January 1990, pp.440-443.

[Tenney and Sandell 81] Tenney R.R. and Sandell N.R., Structures for Distributed Decisionmaking, *IEEE Transactions on Systems, Man and Cybernetics*, vol.SMC-11, no.8, August 1981.

[Thum and Liew 91] Thum P.C., Liew A.C., Computer Analysis of Transmission Line Insulation for Lightning Performance, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.780-790.

[Uma *et al.* 93] Uma G, Prasad B.E, Kumari O.N, Distributed Intelligent Systems: issues, perspectives and approaches, *Knowledge-Based Systems* vol.6 (2) June 1993,pp.77-86.

[Vazquez *et al.* 96] Vazquez E., Chacon O.L. and Altuve H.J., An On-Line Knowledge Based System for Fault Section Diagnosis in Control Centers, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 1996, Florida, USA, pp.232-236.

[Verho *et al.* 95] Verho P., Jarventausta P., Karenlampi M. and Partanen J., Intelligent Configuration Management of Distribution Network, *Proceedings of International Conference on Energy Management and Power Delivery 1995*, EMPD 1995, Singapore, pp.43-38.

[Vickers and McDermid 93] Vickers A.J. and McDermid J.A., An Approach to the Design of Software for Distributed Real-Time Systems, *Technical Report YCS-93-21, 1993*, Department of Computer Science, University of York, Heslington, York.

[Vranes 95] Vranes S., Integerating Multiple Paradigms within the Blackboard Framework, *IEEE Transactions on Software Engineering*, vol.21 (3), March 1995, pp.244-262.

[Wagenbauer and Nejdl 93] Wagenbauer M.P. and Nejdl W., Model/Heuristic-Based ALarm Processing for Power Systems, *AI Edam*, vol.7(1) 1993, pp.65-78.

[Wielinga *et al.* 92] Wielinga B., Van de Velde W., Schreiber G., Akkermans H., The KADS Knowledge Modelling Approach, *Technical Report 1992*, University of Amsterdam, Social Science Informatics, The Netherlands.

[Winter *et al.* 95] Winter R.L, O'Brien J., Wakefield R., A New Schema for the Engineering Design of Autonomous Agents in Non-Manufacturing Domains, *Proceedings of IEA95AIE*, Melbourne 6-8 June 1995, pp.21-26

[Wittig *et al.* 94] Wittig T., Jennings N.R., Mamdani E.H., ARCHON - A Framework for Intelligent Co-operation, *Technical Report 1994*, Queen Mary and Westfield College, London.

[Wooldridge and Jennings 94] Wooldridge M. and Jennings N.R., Agent Theories, Architectures and Languages: A Survey. *Proceedings ECAI-Workshop on Agent Theories, Architectures and Languages*, Amsterdam, The Netherlands, 1994, pp.1-32.

[Wong and Kalam 94] Wong S.K. and Kalam A., Development of a Decision Support System for Power System Protection System Design, Analysis and Assessment, Universities Power Engineering Conference 1994, (*UPEC '94),* University Galway, Ireland, pp.39-42.

[Wong *et al.* 94] Wong S.K., Kalam A., Klebanowski A., A Decision Support System Using Case-Based Reasoning in Power System Protection, Australiasian Universities of Power Engineering Conference 1994, (*AUPEC '94)*, pp.682-687.

[Wong *et al.* 95] Wong S.K., Kalam A. and Leung C.H.C., Incorporating Distributed Problem Solving Technique Into A Case-Based System. *Conference Proceeding IEA/AIE 1995.*

[Wong and Kalam 95] Wong S.K and Kalam A., Development of A Power Protection System using an Agent Based Architecture. *International Conference on Energy Management and Power Delivery*, EMPD 1995, Singapore, pp.433-438.

[Wong *et al.* 96] Wong S.K., Alwast T., Biasizzo C., and Kalam A.,
Agent Based Technology and its application in Power System Protection. to be published in the International Journal of Power and Energy Systems 1997, The International Association of Science and Technology for Development (IASTED), Canada.

[Wong and Kalam 96] Wong S.K. and Kalam A.,
Distributed Intelligent Power System Protection Using Case Based and Object Oriented Paradigms, *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, ISAP 96, Florida, USA, pp.74-78.

[Wong and Kalam 97a] Wong S.K. and Kalam A.,
Intelligent Power System Protection Using Agent Technology, The 31st Universities Power Engineering Conference (UPEC '96) to be held in Technological Educational Institute Iraklio, Iraklio, Greece, 18-20 September 1996, *to be published in the Proceedings of UPEC 1996.*

[Wong and Kalam 97b] Wong S.K. and Kalam A.,
An Agent Approach to Designing Protection Systems, Sixth International Conference on Developments in Power System Protection (DPSP '97) to be held in University of Nottingham, UK, 25-27 Mar 1997, *to be published in th Proceedings of DPSP 97.*

ADDENDUM

Proceedings of the 29th Universities Power Engineering Conference 1994

Imeachtaí an 29ú Comhdháil Ollscoileanna um Innealtóireacht Chumhachta 1994

14, 15, 16, September, 1994

Department of Electrical Engineering
Faculty of Engineering

Volume 1

UCG

Coláiste na hOllscoile Gaillimh
University College Galway

DEVELOPMENT OF A DECISION-SUPPORT SYSTEM FOR POWER SYSTEM PROTECTION SYSTEM DESIGN, ANALYSIS AND ASSESSMENT

*S.K Wong **Dr A. Kalam

*Department of Computer & Mathematical Sciences
**Department of Electrical & Electronic Engineering,
Victoria University of Technology, Melbourne, AUSTRALIA.

ABSTRACT

The proposed system uses integration of different methodologies and techniques viz. case-based reasoning (CBR), causal reasoning, object-oriented concepts. The system stores past designs of protection schemes for busbar, generators, motors, feeder lines and transformers. It is capable of offering explanations, assesses the possible causes of an anomaly, used as a teaching tool and aid engineers in decision-making and in design, analysis and assessment of a protection scheme. An introduction of the system, the proposed methodology, background of the related work, the system's architecture and an example of a part of the system is given.

1. INTRODUCTION

1.1 The Nature of the Research

The nature of the research concerns with the development of a decision-support system in the area of power system protection. The application system aids the protection engineers in the design, analysis and assessment of a power system protection scheme. The design of a power system is tedious work which requires great deal of expertise and experience. The knowledge of a protection engineer is not acquired from texts and reference books. The decision making made by a protection engineer are usually subjective and heuristic.

A power system consists of many components that needs to be protected against occurence of any series or shunt faults. Protective schemes built for a power system depends on a lot of factors. Such factors include the system's requirements, the location and the importance of the power system, the components to be protected, the economic constraints, the availability of resources, the authority's policies, and so on. However, the most important factor when deciding whether to implement a particular protection scheme is economy.

1.2 Design and Assessment

The functions of the application system involve two major activities - design and assessment of a protective scheme. Design is a complicated activity which is always tied to a number of constraints. In practice, most design work do not start on the first principle basis. The process relies not only on knowledge of design styles and domain knowledge such as principles and performance theoretical guidance, but also on the use of previous designs. If the given problem is ill-defined, then previous designs, experience and heuristics play a very important role in the design process. Generally, most designers always relate to their past experiences and may even refer to the work of other designers.

Assessment activitiy involves assessing the current design and checking whether the design covers every aspect of the problem and whether sufficient coverage or attention has been given to each part of the problem. If all the requirements and the constraints of a problem are satisfied, then the design is considered to be good and no further analysis is needed. But if a design fails to meet all or some of the essential requirements, then further evaluation would be required. The final analysis may include suggested solutions, ammendments or modifications of the original design. However, this step often involves references to be made on past designs and experiences. This is deemed important and necessary to justify any modifications made. In other words, both the system's functionality, design and assessment depend very much on past designs and experiences. This approach actually helps to speed up the work and make the process more efficient.

2. THE ADOPTED METHODOLOGY

The requirement of the system is to take in a problem, analyse it and then retrieve the similar cases. The retrieved cases are then used to adapt and transform the problem case into a solution. The system also needs to have a database to store past designs as individual cases.

The development of the system uses integration of different methodologies and techniques. The main techniques are case-based reasoning (CBR), causal reasoning and object-oriented concepts. A number of other methodologies which includes petri-nets, blackboard architecture and distributed artificial intelligence has been studied. However, these methodologies are more applicable for systems which models distributed artificial intelligence and the communication among these agents. Hence, it appears that the proposed methodology seems to be the most applicable and suitable approach for the development of this system.

Case-based reasoning is one that recalls (retrieves) cases of past experiences from the memory base that are similar to the current problem case and solves or interprets a problem by reasoning with past solutions [1]. Human experts solve a problem in their area of expertise, especially in domains such as law, mathematics, design and strategic planning rely heavily on memory of past cases or experience.

The application of CBR areas are in architecture design [4], mechanical design [5, 6] and also in the diagnosis and therapy suggestions for cardiac disease patients [7] and management of faults in communications networks [8].

Case-based reasoning addresses the issues of analyzing, representating, organizing and retrieving records of past experiences. Hence, case-based reasoning is said to be an important method of problem-solving and reasoning. Using CBR, the system resolves a problem by retrieving similar cases from the case library and adapting the problem case into a reasonable and acceptable solution.

Sometimes in a CBR system, no similar cases could be retrieved from the case memory or the retrieved cases could not be used to adapt and transform a problem case. At other times, a problem may lack essential information. In both situations, causal reasoning is used. In causal reasoning, knowledge domain is used to construct an explanation of their reasonings. In addition, this reasoning could also be used as part of an evaluation program of the system to explain why failures occur in some circumstances.

The application of object-oriented concepts and programming could help with the development of a more efficient, flexible and portable system. Object-oriented database systems eg. ODE (object database and environment) and O_2 system[1] [15] provide a number of facilities that could alleviate programming and enable better systems to be built. Both are database systems and environments based on the object paradigm. They provide facilities for creating and manipulating persistent objects and different versions of an object, and associating constraints and triggers with objects.

3. RELATED WORK IN POWER SYSTEM

There have been quite a number of utility systems developed in power system lately. To quote an example of such importance of these utility systems, [2]the Electrical Systems Division of Electric Power Research Institute (EPRI) in 1984 - 1988, has committed a majority of its financial resources to the development of electric utility systems in matters concerning power system economies, efficiency, reliability, operations and the environment. Priorities and funding of the Division programs were placing added emphasis on projects which will prolong the life of existing equipment and result in more efficient system operation. Some of their goals are to extend the useful life of existing system, demonstrate promising "next generation" technologies and develop improved analytical and design techniques. The technical approach is to develop advanced tools and techniques for system planning, analysis and design.

More recently, in an international survey [2] carried out on power system applications, there was greater emphasis placed on the development and application of expert system in power system which uses more sophiscated approaches to knowledge representation. It emphasized that there is a considerable amount of work to be done on the development of a systematic methodology for knowledge acquisition in the power systems area, which is identified as a 'bottleneck' in developing these systems.

The survey [2] also revealed the potential applications of expert systems to power systems. This includes system operation, system control, system restoration, system planning/design, substation automation, maintenance scheduling and alarm processing. The following items have been recognized as appropriate areas and needs for expert systems to be applied to power systems: identifying alert or emergency states, providing emergency procedures, system planning, operator training.

Protection for power systems is a very large and important area that requires expertise in the management of changes and complexity, especially more so, whenever a problem arises. The power plant consists of a lot of expensive elements. A fault or any abnormality can be intolerable and unaffordable as well as it can be extremely dangerous and risky, not to mention the fact that a high cost will be incurred for the repairs too.

There have been an increase in the number of expert systems developed in protection areas such as alarm handling and fault diagnosis [9, 10, 11] and protective relaying [12, 13], monitoring [14].

4. THE ARCHITECTURE

The design of the architecture to complement the methodology is given in Figure 1. The architecture consists of seven modules or components: a *Planner* module that contains set of plans, a *Builder* module which is responsible for building a problem case, an *Inquirer* module that stores sets of queries - each of the query set being associated with a specific plan, an *Adapter* module, an *Explainer* module, Graphical User Interface (*GUI*) module and the *Database* itself.

The *Planner* contains a set of pre-defined plans but only a single plan can be accessed at any one time. Each plan is directly associated to one specific type of problem. The input to this module would be from the user stating the type of the problem that needs to be solved. This means that a plan retrieved as an output of this module will depend on the type of the current problem. The *Builder* module is responsible for constructing a problem case, ie. instantiating an object from the specifications of the problem case. The problem case consists of information received from the user. This module receives a plan as its input from the *Planner* module. This plan has a set of formulated questions for the user. The response from the user provides the necessary information and details of the problem case. It is very important to have an accurate information and proper understanding of the problem to help with the retrieval of the most appropriate cases. Once the problem case is created, the *Inquirer* module is invoked. Using the problem case as a base, an appropriate set of queries is selected from this module. This set of queries is performed on the database to retrieve similar cases.

[1] O_2 is a system used for the design and development of object-oriented database system ideal for developing large-scale client/server applications. Programming can be done using C, C++ or O_2C language. It also has a SQL-query language that is used to query the database.

[2] EPRI 1984 - 1988 Research and Development Program Plan.

The *Database* itself is the case library. Each case in the system is represented as a complete solution itself. However, a case can have references or pointers to other cases. Maintaining a reference or a pointer in this manner provides a cleaner structure especially as regards the storage and manipulation of objects. For example, busbar cases have pointers to bitmaps and to relay case as well. All interactions between system and the outside world, ie. the users take place in *GUI*. *GUI* is built as a friendly user environment where users of the system especially the novice and the inexperienced users can operate the application system with minimum or no training at all.

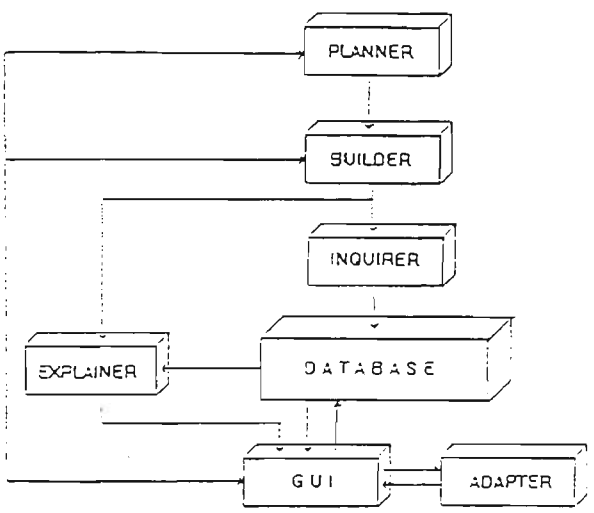


Fig.1 The system's architecture

The *Explainer* module takes control over the processing whenever CBR fails. CBR may fail when there is insufficient essential information contained in a problem case or when there are no retrieved cases. The *Explainer*, using the knowledge domain provide information and explanation to solve a problem.

The *Adapter* module uses the problem case as the basis for any modifications that the user may have. This action is not automatic. The *Adapter* module is invoked only if the user wishes to adapt the problem case and transform it into a new solution case which is then stored in the *Database*.

5. THE APPLICATION SYSTEM

The application could be used as a teaching tool or as a training ground for the new and inexperienced graduate engineers. It provides the much sought and needed information that are not easily available from textbooks and reference materials or journals. Furthermore, the development of the application also aids the engineers in the design, analysis and assessment of a power system protection scheme especially at times when the experts are not freely available. It could also be used to assist the inexperienced engineers in decision-making.

The proposed type of program mentioned has been identified and recognized as a priority area by the authorities and other interested bodies concerned. For example, State Electricity Commission of Victoria, Electricity Supply Association of Australia, the Department of Electrical and Electronic Engineering at VUT and other organizations, both at national and international level, has considered this work ***urgent*** and ***necessary***[3]. This program will fulfil the need of any utility's power system protection division.

The expert system applications to power system problems is a relatively new area. According to the international survey [2] done on power system applications, most of the expert system projects are in the stage of idea and prototype only, very few have developed for practical operational stage. The objectives of the expert systems have been broadly classified into six categories: planning, monitoring, control, system analysis, education, simulation and others. While monitoring has been identified as the biggest category, it was recognized also that application areas such as planning, educator, simulator and analysis lack the equal attention. Hence, strong needs for expansion into this specified areas are being called for [3].

6. DESIGN OF A PROTECTION SCHEME FOR A POWER SYSTEM

A power system could be divided into different parts or components of a power system. The protection of a power system could be seen as the sum of protection of all parts of the power system. The major parts of the power system to be protected are busbars, generators, motors, transmission and feeder lines, and power transformers. Each of these components has different applicable protection schemes depending on the requirements of the system, the importance and the location of the power system, the cost of the protection scheme, the availability of resources, the authority's policies and so on.

Figure 2 illustrates the flow chart of the application system which designs a protection scheme for any part of a power system. The proposed system starts with an enquiry to the user to find out the particular type of problem that needs to be solved. That is, which component of the power system needs to be protected, eg. busbar, motor, generator, transmission & feeder lines or transformer.

Once the problem type is known (eg. busbar protection), the system will trigger the Planner Module to retrieve the appropriate plan. (In this example, the retrieved plan is associated with protection for busbars.) The execution of the plan requires interactions with the user. Information received from the user is used to create and instantiate an object, which is referred to as a problem case. The problem case contain not only text information but also equations needed to calculate the settings for a certain type of relay and also a pointer to a bitmap file which stores the configuration diagram of the component.

Analysis of the problem case is done in the Inquirer module. Analysis is carried out so that the correct and proper queries could be performed on the database. These queries will determine the efficiency of cases retrieved. Similar cases will be retrieved and presented to the user together with the problem case for consideration. The user is given an option whether he/she wants to adapt the

problem case. The problem case can then be modified and used as a basis for developing a new solution case.

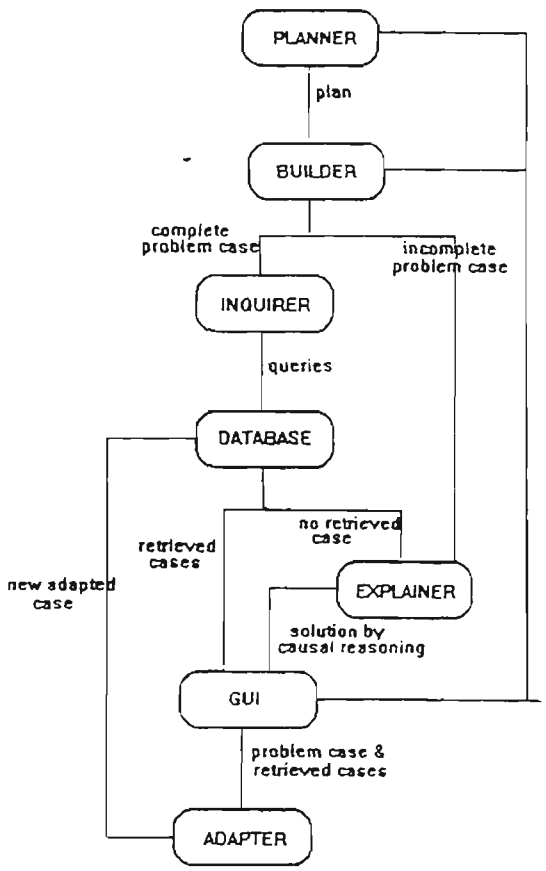


Fig.2 The system's flow-chart

7. CONCLUSION AND FURTHER WORK

The methodology presented appears to be most suitable for the development of the system. The integration of different techniques, ie. CBR, causal reasoning and object-oriented concepts enables a more robust and efficient system to be built. A prototype of a small part of the proposed system has been built. The results have been promising.

At this immediate stage, work is well under way to transfer and develop the prototype in O2 system. Development work would be extended to implement the complete system. Besides that, further research includes other issues like efficient retrieval and storage of cases.

ACKNOWLEDGEMENTS

The authors are grateful to the Australian Electrical Supply Industry Research Industry Board (AESIRB) for financial assistance.

REFERENCES

[1] Riesbeck, C.K and Shank, R.C : "Inside Case-Based Reasoning",1988.

[2] International Survey of Power System Applications, Expert Systems Applications in Power Systems, Prentice Hall 1990.

[3] Power System Protection - Short Course Program, Electric Supply Association of Australia and Department of Electrical and Electronic Engineering, Victoria University of Technology, 1993.

[4] Domeshek E., Kolodner J. : "Using The Points of Large Cases", AI EDAM, vol.7 (2), 1993, pp. 87-96

[5] Tanaka T., Hattori M., Sueda N. : "Use of Multiple Cases in Case-Based Design",The Eighth Conference on Artificial Intelligence for Applications,Monterey, California,Mac 2-6, 1992.

[6] Bardasz T. ,Zeid I. : "DEJAVU: Case-Based Reasoning for Mechanical Design", AI EDAM, 1993, 7(2), pp.111-124

[7] Koton P.: "Reasoning about Evidence in Causal Explanations", Proceedings Case-Based Reasoning Workshop 1988, pp.260-270.

[8] Lewis L.: "A Case-Based Reasoning Approach to the Management of Faults in Communications Networks", IEEE, Conference on Artificial Intelligence for Applications 1993.

[9] Eickhoff F., Handschin E., Hoffmann W.: "Knowledge Based Alarm Handling and Fault Location in Distribution Networks", Power Industry Computer Application Conference 1991, IEEE, pp.358-364.

[10] Sugihara H.: "A Practical Expert System for Indicating Faulty Sections withing a Control Center", Expert System Applications to Power Systems IV, 1993, CRL Publishing Ltd., pp.236-241.

[11] Ypsilantis J., Yee H.: "Genetic Learning Applied to Distribution System Fault Diagnosis and Alarm Processsing", Expert System Applications to Power Systems IV, 1993, CRL Publishing Ltd., pp.393-397.

[12] Khaparde S.A., Kale P.B., Agarwal S.H.: "Application of Artificial Neural Network in Protective Relaying of Transmission Lines", Applications of Neural Networks to Power Systems - First International Forum-Papers, IEEE, 1991, pp.122-125.

[13] Kawahara K., Sasaki H., Kubokawa J., Sugihara H., Kitagawa M.: "An Expert System for Supporting Protective Relay Setting for Transmission Lines", Developments in Power System Protection, IEE, 1993, pp.203-206.

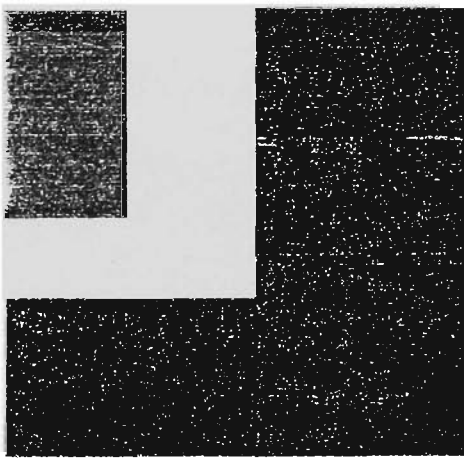
[14] Fauquembergue P., Maizener A., Parant J.M., Perrot L., Bertrand H.: "Monitoring of Protection System Behaviour using an Expert System which Analyses Substations Sequential Events Recording", Developments in Power System Protection, IEE, 1993, pp.42-45.

[15] O2 Tools - User Manual (version 4.3), O2 Technology, 1993.

AUPEC'94

Australasian Universities Power Engineering Conference

Adelaide - Australia
27-29 September 1994



Proceedings
Volume 3

Edited by
Özdemir Göl

UNIVERSITY OF SOUTH AUSTRALIA

A DECISION-SUPPORT SYSTEM USING CASE-BASED REASONING IN POWER SYSTEM PROTECTION

*S.K. Wong **Dr A. Kalam ***Dr A. Klebanowski

*Department of Computer and Mathematical Sciences
**Department of Electrical and Electronic Engineering
Victoria University of Technology, Melbourne, Victoria, AUSTRALIA.
***Control and Protection Resource Group
National Electricity, Melbourne, Victoria, AUSTRALIA.

Summary

A decision-support system for designing, analysing and assessing protection schemes for a power system is introduced. The system consists of an innovative application and includes the following features:

- utilizes and integrates different reasoning methodologies including case, rule and explanation-based methods;
- encapsulates knowledge as well as data in an object-oriented environment;
- stores previous protection scheme designs in an object-oriented database and retrieves by query;

Although the primary purpose of the system is to assist protection engineers in the design of protection schemes, the system could also be used in the training of graduate engineers.

1. Introduction

There has been an upsurge in interest in expert system applications in power systems since 1981 [10]. Moreover, the number of expert system applications in power system protection has been increasing steadily in the recent years. Most applications have been in the areas of alarm processing and fault diagnosis, steady-state and dynamic security, planning and design, system restoration, environments for operation aids, remedial controls, substation monitoring and control and maintenance scheduling [1]. Also a number of simulation programs and knowledge-based systems have been developed as learning tools and consultation systems.

Although the research interest in the application of expert systems in power system protection has been growing, there is still not enough importance and emphasis given to the development of decision-support systems in this area. Expert systems are merely automated systems which derive a conclusion from a set of production rules. This is usually achieved by applying a forward or a backward chaining process. Decision-support systems, on the other hand, play more of a support role and may involve frequent user interaction. A decision-support system assists the decision-maker in selecting the most appropriate decision or course of action. It may offer a number of alternative solutions or the best solution given the problem constraints. On the request from the user, the system may examine and evaluate the different options available to the decision-maker. Decision-support systems are most successful in application domains where decisions have a

more subjective nature and are based on heuristics and experience rather than on well-defined algorithms or analogical reasoning.

This research is aimed at developing a decision-support system in power system protection. More specifically, the system is intended to assist protection engineers in the design, selection and analysis of protection schemes for a power system. In addition, the system could also be used as a learning and teaching tool for new graduates and inexperienced engineers. The system assists users in the selection of appropriate protection scheme subject to certain requirements and constraints. The system addresses the problem by retrieving similar designs from the database. The user is then asked to decide whether to adopt or adapt the retrieved designs.

2. Application of Expert Systems in Power System Protection

One of the main advantages of expert systems is the retention of knowledge when a human expert retires from the field [10]. Lai [2] has developed a prototype expert system for power system protection coordination. The system records and models the knowledge of power engineering experts. The output of the system is a design scheme for protection coordination of a power plant that meets the regulatory requirements. It was concluded that the system has proved to be successful but the knowledge base needs to be expanded.

Examples of other systems developed include a general-purpose fault simulation program [3] for detailed modelling of faults on a high voltage overhead transmission line, an expert system [4] to perform the setting, coordination and overreach resetting of protective distance relays in high voltage transmission networks and a system [5] that performs fault analysis and alarm handling in a distribution network.

Lu et. al. [6] has developed a knowledge-based tutorial and consultation system designed to assist the control center operators and Energy Management System software integration team members. The system has been implemented on a personal computer. It was concluded that with appropriate design, the system could become an excellent tutor for knowledge transfer.

An expert system, SEPT [7] was developed for monitoring of the protection system behaviour. The system has been developed using an object-oriented language and a set of production rules. It performs exhaustive checks on the operation of protection equipment and detects any inconsistencies in logic and time. It then estimates the location of the faulty section and attempts to classify the incident.

Kawahara et al. [8] developed an expert system designed for setting the directional overcurrent and distance relays with regard to loop systems. The knowledge of the relay setting standards is translated into production rules and linked with numerical computations.

An interesting computer-aided protection engineering system, CAPE, [9] has been developed recently, and is still undergoing development. The system has a range of capabilities including computation of short circuit currents, relay checking, event-stepped simulation of relay and circuit-breaker operations and planning of power flow and transient stability studies. It consists of a general system protection relational database, a full-screen database editor and eight modules for analysis and reporting that provide a comprehensive computing and record-keeping environment.

Application of CBR in Power System Protection

Most of the expert systems developed in the protection area employ production rules [7,8]. Rule-based systems, ie. systems that employ production rules, are easy to implement and simple to use but usually involve tedious programming. The rules have to cover all possible cases which may arise in the problem domain. In addition, rule-based systems work well in areas where the domain is well-defined. On the other hand, in application areas where the problem domain is not well-defined and requires heuristics, experience and plain common sense, the rule-based systems do not work [11]. Other reasoning techniques have to be employed and case-based reasoning (CBR) is one paradigm that has proven effective in many experimental and applied systems. CBR "remembers" previous problems and uses them to solve and evaluate current problem as well as to adapt previous cases to modify and transform the problem case into a new case [12].

The number of research projects using CBR has increased prolifically in the past few years. The areas of applications have grown and the systems that have been developed include those which use cases to resolve disputes, design, planning, legal reasoner and diagnosis [11]. Most of the case-based systems developed were reported to be successful in replacing rule-based systems too.

The design and selection of protection schemes for various parts of the power system requires not only domain knowledge but experience and skill as well. An expert may be good in analogical reasoning but the expert may not be good in remembering. When a new problem is encountered, it is quite natural to investigate past but similar problems and try to either adopt or adapt the previous solution(s) to the current situation.

The system will assist the protection engineers especially as regards the remembering of previous designs. The system uses case-based reasoning (CBR) technique. CBR technique builds a case library consisting of previous designs and retrieves cases which are similar to the current problem. The user could either adopt or adapt any of the retrieved case(s) as a possible solution to the current problem. If no relevant cases exists in the case library, the system will automatically generate a solution to the user's problem. Evaluation and verification of the solution, that is, settings of the relays are then carried out.

In addition to CBR technique, the system uses object-oriented design and database, and other reasoning techniques like rules and explanation-based. Building a system that uses an integration of different methodologies and techniques adds to the efficiency, performance and robustness of the system. For example, if there are no cases in the case library, the system could utilize other reasoning techniques such as causal, functional, evidential, rule and explanation-based which are incorporated into the system.

4. Proposed Methodology

All entities in the world can be viewed as objects. For example, people, buildings, files, etc. can be regarded as objects which are related to and interacts with each other in a variety of ways. Any objects can be easily represented and neatly modeled using object-oriented technique. Thus given an object, the features and behaviour associated to the object can be easily described.

The above viewpoint has been adopted when developing the system. It is very natural to implement cases as objects in a case-based system. The cases in the system includes not only data or features and information about the protection schemes but knowledge as well. A problem case in the protection system has the capability to deduce the appropriate protection schemes based on

the data and information contained within, using case-based and rule-based reasoning techniques. It can also explains why the protection schemes were selected. These objects are known as knowledge sources (ks). A ks is a sophiscated data structure with an intelligent or expert system built-in. In other words, it is a knowledge-based entity with an inference engine.

CBR is employed as the main reasoner to retrieve similar cases and apply the solution of the retrieved cases to the current case. The advantage of using CBR is that the system does not need to reason from first principle which involves tedious programming. Besides, CBR is a very natural way of reasoning that people employs in their daily decision-making process.

Similar cases are retrieved from the case library by queries. Queries are formulated and performed on the case library based on the features of similarity that it is looking for from the current case. Query processing is an easy and efficient way of retrieval compared to retrieval by rules. However, if CBR fails to retrieve any cases or the retrieved cases are unacceptable to the user, the system goes through a rule-based reasoning process. Rules may be one of the simplest and most common reasoning among other reasonings technique but it still proves to be powerful in many situations as well.

Before presentation of the deduced solution, the system goes for a second retrieval process. If similar cases are found, they are offered as the suggested solutions. If the user do not accept the retrieved cases or no similar cases are found, the system will perform some computation and verification processes before presenting the deduced solution. The user can also query the system as to why the protection scheme was selected. Futhermore, new information can be updated into the cases. This capability of the case-based system is different to the adaptation process. Adaptation is a process that modifies and transforms the retrieved cases into a new case which is then saved in the database.

5. The Application System

The development of the system requires a lot of discussions and interviews with the protection engineers and experts. This collection and acquisition of knowledge and information are needed to build the knowledge-base and the overall system. Emphasis is given on their expertise and their approach in dealing and solving a problem in order to build an applicable system. This is to prevent the application system to be too theoretical focused or based which could results in the development of another prototype system. A prototype system would validate the proposed methodology but would not be practical to be used and applied in the real world.

5.1 An Illustrative Example of the System

A part of the application system has been implemented and undergoing refinement and further development. This part of the system involves busbar protection (figure 1). The system begins by asking the user some fundamental questions, eg. whether remote protection is required, whether dedicated current transformers (cts) are available, etc.

From the user's response, the system would start its first retrieval. If the retrieval is successful, the retrieved cases are presented to the user. If the solutions of the retrieved cases are acceptable to the user for application to the current problem, then no further questions would be asked. Otherwise, the user is given the option to either adapt the retrieved cases to the problem case or let the system resolves a new solution altogether. If the latter option is taken, further information is required with regards to the busbar. This includes minimum and maximum fault levels, load

current and information of each cts used, eg. magnetizing current level, knee point voltage level, etc. All these data collected will build the problem case.

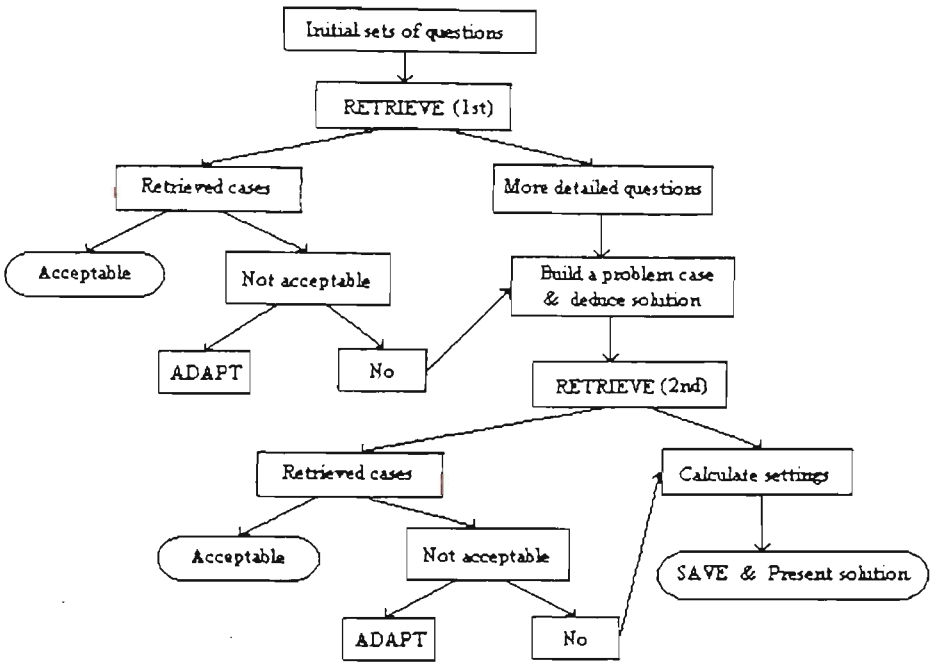


Figure 1. The system's flowchart

If no cases are retrieved, the system goes through a reasoning process to infer an appropriate protection scheme. Upon completion, the system attempts a second retrieval based on the inferred protection schemes. However, if this process fails to retrieve any cases from the database, the system then compute the relays' settings and presents the recommended protection scheme inferred. This forms the new case which is then saved into the database.

The cases in the database keep complete information, requirements and design of a bus. However, the relays that are '*attached to*' the protection scheme are treated as individual cases. They are '*linked dynamically*' to the bus case. This '*dynamic binding*' gives an added advantage whenever any new relay is added to the database. For example, consider a new high impedance relay which is recently added to the database. Any future retrieval of bus cases of high impedance protection scheme will retrieve all the high impedance relays in the database including the latest addition. A date is kept on the bus and relay cases so that the user can differentiate between the updated relays from the 'original' relays that were applied to the bus concerned.

This feature adds flexibility and efficiency to the CBR system. It prevents duplication of information in the database. In addition, memory space could be utilized efficiently because only one copy of a relay case is kept. This would not be the situation if a bus case contains the protection schemes with the relays attached to it '*statically and permanently*', which would be the situation if the current normal practise of case storage in case-based systems is adopted.

6. Conclusion and Future Work

The part of the system built gives a positive and promising result of the proposed methodology. Further work involves research into issues like storage, possibility of other reasonings be employed in the explanation and adaptation processes, and development of other parts of the application system.

7. Acknowledgement

The authors are grateful to Charles Biasizzo, Protection Consultant, Bruce Bennett, Protection Engineer of National Electricity, Melbourne for their design contributions and expertise in the development of the system and Ted Alwast, Computing and Mathematical Sciences Department, Victoria University of Technology, for his valuable comments. The authors are also grateful to Australian Electricity Industry Supply Research Board funding for this project.

Reference

1. C.C. Liu, T.K. Ma, K.L. Liou, M.S. Tsai, Practical Use of Expert Systems in Power Systems, ESAP IV, Melbourne, Australia, 4-8 January 1990, Plenary paper.

2. L.L. Lai, Development of an Expert System for Power System Protection Coordination, Fourth International Conference in Power System Protection, 11-13 April 1989, pp.310-314.

3. A. Kalam, C. McLean, Transient Fault Analysis of Faulted Overhead Lines and Their Associated Protection, Fifth CEPSI, November 1984, pp.19-23.

4. M. A. Madkour, M. A. H. El-Sayed, A Knowledge based approach for Setting Protective Relays in Transmission Networks, Electric Power Systems Research, 27 (1993) pp.107-115.

5. F. Eickhoff, E. Handschin, W. Hoffmann, Knowledge Based Alarm Handling and Fault Location in Distribution Networks, 1991 Power Industry Computer Application Conference, IEEE, pp.358-364.

6. C.N. Lu, K.C. Liu, M.R. Unum, A Knowledge Based Consultation Facility for Power System Application Software, Ninth CEPSI 1988, pp.359-368.

7. P. Fauquembergue, A. Maizener, J.M. Parant, L. Perrot, H. Bertrand, Monitoring of Protection System Behaviour using An Expert System which Analyses Substations Sequential Events Recordings Field Experience at Electricite De France, Fifth International Conference on DPSP 1993, IEE Publications, pp.42-45.

8. K. Kawahara, H. Sasaki, J. Kubokawa, H. Sugihara, M. Kitagawa, An Expert System for Supporting Protective Relay Setting for Transmission Lines, Fifth International Conference on DPSP 1993, IEE Publications, pp.203-206.

9. M.K. Enns, P.F. McGuire, R. Ramaswami, A. Arbor, CAPE: The Computer-Aided Protection Engineering System, Proceedings of the American Power Conference, v.2, 13-15 April 1992, pp.1084-1089.

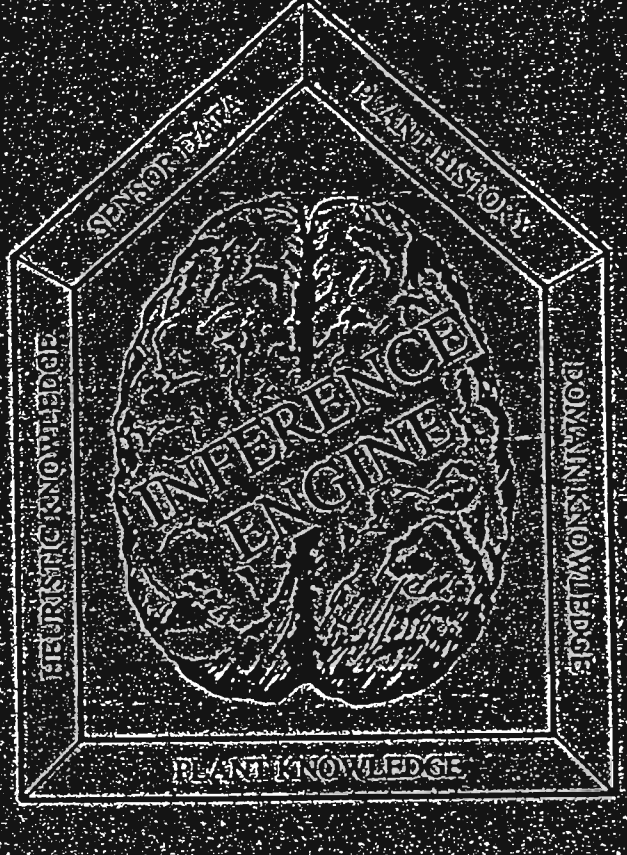
10. K.L. Lo, I. Nashid, Expert Systems and Their Application to Power Systems: Part 1 Components and Methods of Knowledge Representation, Power Engineering Journal, February 1993, pp.41-45.

11. J.L. Kolodner, Improving Human Decision Making through Case-Based Decision Aiding, AI Magazine, Summer 1991, pp.52-68.

12. J.L. Kolodner, W. Mark, Case-Based Reasoning, IEEE Expert, October 1992, pp.5-6.

INDUSTRIAL AND ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

IEA/AIE 95



Proceedings of the Eighth International Conference
Melbourne, Australia, June 6–8, 1995

Edited by

Graham E. Tweed & Moonis Ali

Gordon and Breach Publishers

INCORPORATING DISTRIBUTED PROBLEM SOLVING TECHNIQUE INTO A CASE-BASED SYSTEM

*S.K. Wong **A. Kalam *C.H.C. Leung
*Department of Computer and Mathematical Sciences
**Department of Electrical and Electronic Engineering
Victoria University of Technology
Victoria, AUSTRALIA.

ABSTRACT

This paper presents an object based architecture that uses cooperating and distributed agents in a case-based framework. The proposed architecture incorporates agents into a case-based system to achieve better coordination. In this paper, an agent corresponds to a knowledge-based entity. This architecture consists of set of entities and agents organised in a federated structure, which are managed and coordinated by the facilitator and controller modules respectively. This architecture is applied in the development of a case-based system to design, analyse and assess power system protection schemes (CAPP). Part of the system has already been implemented and the results obtained thus far indicate that the proposed framework appears to offer substantial advantages over conventional approaches.

1. INTRODUCTION

Distributed artificial intelligence (DAI) is a subarea of AI which is also referred to as cooperative distributed problem-solving (DPS) [1]. DPS is concerned with the application of AI techniques and multiple problem solvers (ie. agents) [2]. It involves distributing control and data to achieve cooperation, coordination and collaboration among the agents which are necessary to solve a given problem. Most of the agents in a distributed system has sufficient knowledge to generate at least a partial or a sub-solution. Therefore, cooperation or coordination among the agents is inevitable and necessary to solve a problem.

The most important feature of DPS that helps to achieve results in the presence of complexity and uncertainty is cooperation. Cooperation between problem solvers or agents must be structured as a series of carefully planned exchanges of information [1]. Performance also depends on the problem solving architecture [1]. Therefore, it is appropriate to consider frameworks or strategies for cooperation. Examples of some of these frameworks are contract net, blackboard model, distributed, parallel blackboard models, scientific community metaphor and organisational structuring [1].

As stated by Decker [2], one of the most powerful aspects of AI approach to problem solving is the ability to deal with uncertain and incomplete information. Therefore, the approach outlined here proposes the integration of DPS and case-based reasoning (CBR) techniques. One of the many advantages of CBR systems is that it can handle incomplete information or missing data quite well [19]; that is, the features of the missing values will not be used in the retrieval process. The retrieved cases which possess a variety of values for the missing feature(s) can determine its influence and importance on the solution proposed. If the solution or outcome of the retrieved case(s) is acceptable, then the missing value has proven to be unimportant. Otherwise, a new solution can be derived using some other reasoning technique.

This paper presents a system that uses cooperating and distributed agents in a CBR framework. This system, CAPP is a case-based system being developed in the area of power system protection for the design, analysis and assessment of protection schemes. This innovative application system represents agents as CBR cases. The system's architecture exhibits the working model of the framework. The paper is organised as follows: section two gives a brief overview of DPS systems, section three gives an introduction to the application system and the design, followed by descriptions of the agents and the system's modules. Section four presents the architecture of the system with a brief illustration on the implementation given in section five. Lastly, section six outlines the conclusion and the directions for future work.

2. A BRIEF OVERVIEW OF DISTRIBUTED PROBLEM-SOLVING SYSTEMS

The need for cooperation and communication between disparate knowledge-based systems has prompted research into the field of DAI. A number of paradigms have been proposed, including the agent-based and blackboard architectures. Khedro et. al [7] introduced an agent-based framework for the development of integrated facility engineering environments in support of collaborative design. This system uses a federation architecture where the agents surrender their autonomy to the facilitators. The knowledge bases and the information are distributed among the collaborating design agents and there is no central database. The agents communicate through facilitators which allow the agents to register and deregister at any time without affecting other agents in the environment.

In DARES [8], a distributed automated reasoning system, the agents are built with incomplete knowledge about the state of the world. A cooperation strategy which is dependent on the initial knowledge distribution was developed to coordinate the semi-independent agents.

Communication in a system of distributed problem solvers (agents) environment is commonly achieved either by message passing or use of blackboard. The AGENTS system developed by Huang et. al [4] achieves communication by both means. AGENTS consists of cooperating expert systems in concurrent engineering design. Emphasis is placed on demonstrating distributed knowledge representation and cooperation strategies for communication, collaboration, conflict resolution and control.

In another multi-agent system [5], the agents plans are taken as constraints because of their inconsistencies with one another. To resolve this conflict, the plans are integrated and modified using a Predicate/Transition (P/T) net. The net represents the total knowledge held by all the agents to achieve their individual respective goals.

Devapriya et. al [6] developed a distributed intelligent systems for FMS control using objects modelled with Petri-nets. The objects built as communicating Petri nets, with object oriented interpretation, are organised into problem solving nodes. Communication among the nodes is achieved via message passing.

Shaw et. al [20] describe two implementation examples of decision support systems (DSS's) for aiding group problem-solving situations. The first system NEST, networked expert systems testbed, is a prototype system, which consists of four expert systems. The second system describes a group decision support system (GDSS) for design of fusion system. Both systems illustrate a multi-agent problem solving system based on the blackboard architecture. The expert systems communicate via a blackboard or mailbox for coordination.

CAPP attempts to take advantage of the benefits of the different techniques described in the above systems. It is built in a CBR framework which utilises the federation architecture. While communication among CAPP agents is achieved via the facilitator, the coordination is, however, managed by a controller module. CAPP agents are built with incomplete knowledge but with the capability of using collected data and information, its knowledge base, and inference engine to obtain a partial solution.

3. CAPP - THE APPLICATION SYSTEM

Protection for power system is seen as a sum of coordinated protections for all parts of a power system which include protections for busbars, generators, motors, lines, transformers, etc. The application system aims to help the protection engineers to analyse and assess a power system or a part of it and recommends an appropriate protection scheme for the power system. Designing a protection scheme for a power system is not an easy task. It is based on a number of factors such as the requirements and constraints of the power system, and also on factors external to the power system. Such factors include economy, authority's policies, location of the power system and etc. Therefore, the decision made by the protection engineers is subjective and based on heuristics and experience.

Protection engineering is the skill and experience of selecting and setting the relays and other protective devices to provide maximum sensitivity to faults and other undesirable conditions. At the same time, the protective schemes have to achieve objectives such as reliability, security, selectivity, speed of output, simplicity and economics. Most of the work and exercises involved are not straight-forward. They require heuristics, experience and common-sense knowledge, which cannot be acquired from any texts or reference books. In addition, the supply of protection engineers in the market place is decreasing. This problem of expertise shortages is expected to worsen during the recent massive restructuring of the electricity supply industries. Furthermore, there is an additional new strain on the protection staff due to recent losses of highly qualified staff and the introduction of new philosophies and technology in the protection and other interrelated areas such as communication, control and system management. Hence, a decision-support system, CAPP aimed to ease the aforementioned problems is instigated.

3.1 System Design

The proposed methodology for the development of CAPP system is outlined in the reference 11. The paper states why CBR has been chosen for this application. The system is basically built up of libraries containing past design cases. Each case contains information about the design and its specifications. The cases, represented as objects using object-oriented concepts, are the agents in the system. They encapsulate not only data and information but knowledge as well.

The operation of the system is controlled by several modules. They are the initiator, the facilitators, the agent-generators, a modifier and a controller. The system has several components, each of which represents one part of the power system where protection is required. Each component can be viewed as a community that houses a specific group of agents. There are no intra- or inter-communication among the agents. Communication is via the facilitators as in federation architecture [7] and coordination is achieved by the system's controller. The following subsections presents a more complete description of the agents and modules of the system.

3.2.1 CAPP agents

CAPP agents are embedded in an object-oriented environment and communicate using an object-oriented language. Each agent is a knowledge-based system with an inference engine and a knowledge base. In other words, an agent is not only a sophisticated data structure but also an an expert system. The agent architecture shown in Figure 1 depicts the four main components which embodies a CAPP agent. The purpose of the interface layer is to smooth out the communication between the agent and the facilitator. The data collected from various sources is used to build the data area component. This component contains the current information and the design specification. The domain knowledge is maintained in the knowledge-base. This knowledge is used by the inference engine to construct a solution which is then stored in the solution base. This base has a number of pointers or references to a database where detailed information is stored. This feature provides the CAPP agent with a rich source of up-todate information, eliminates data duplication and encourages a more efficient memory utilisation. The explainer component provides the justification for the proposed design scheme.

As stated in reference 10, these agents can also be viewed as dichotomous entities having two roles: (i), as a repository case which stores all the features and information about the design and (ii), as an integrating and cooperating entity in the system. The agents are categorised into different communities. Each agent belongs to one community only. An agent is only created or generated if there does not already exist a similar agent in the community. The communities in the system are the database areas which are represented locally and globally. Agents of the same type are found in a local community whereas in a global community, groups of different agents can coexist.

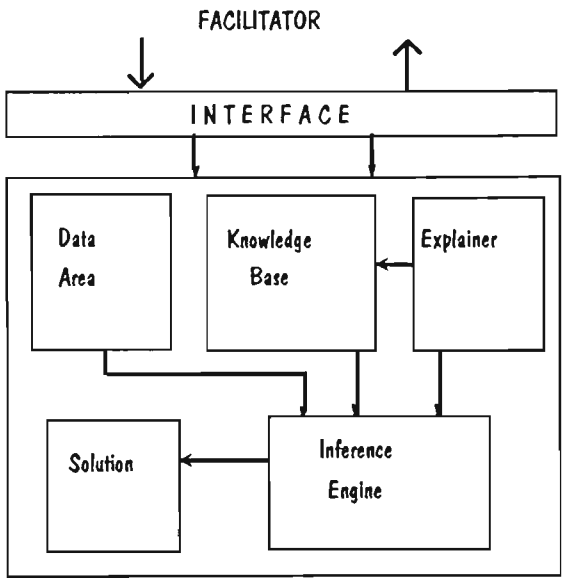


FIGURE 1. Agent Architecture

3.2.2 Facilitators

In this system, one facilitator is assigned to one community and they are invoked by the initiator module. The concept of the facilitators in the system is similar to that in a federated architecture [6], but their functions are quite different. Here, each facilitator functions as a supervisory agent. The facilitator is responsible for scheduling the agent-generator activities and acts as a communication buffer. It communicates the initiator's request to the community. The request is a message listing the user specifications of a power system and the requirements for a protection design. Based on this message, the facilitator selects the agents, if any, from the community that satisfy the required specifications .

3.2.3 Agent-generators

The responsibility of the agent-generators is to generate and build new agents, when there are no existing agent from the community that meets the requirements of the specifications. This process increases the population of a community. One agent-generator is assigned to each community and is invoked by the facilitator.

3.2.4 Modifier

There is only one modifier in the entire system and it resides in the global society. This module is only invoked when agents can collaborate but cannot coordinate with other agents. Therefore, some modification is needed to alter the values of the specification. This process can give rise to a new agent if the modified agent becomes too different from its original form/specification. In other words, the modifier may influence the population increase in a community.

3.2.5 Controller

There is only one controller which acts as a manager in the entire system. The controller is responsible for the organisation of and communication among the agents and controls the activities of the modifier. The primary task of the controller is to coordinate the different agents to construct a solution for the user. The solution represents the design of a protection scheme for a power system based on the constraints given by the user.

However, if the user is only interested in a partial solution which can be provided by one agent, then coordination is not required. Therefore, in such circumstances, the controller will not be required. The function of the local facilitator alone will produce the output required by the user.

4. THE ARCHITECTURE

The design architecture of CAPP was based on the application requirement and the system description. The multiple and distributed problem-solving agents described in this paper exist within the one homogeneous system. As stated earlier, the agents do not communicate with each other directly. Instead, they communicate via their local facilitators and their actions are coordinated by the controller to achieve a common objective.

The messages communicated by the agents represent the sub-solutions, ie. the design information based on the constraints and specifications provided by the user. It is the responsibility of the facilitators or the controller to coordinate, correlate and present the design information to the user. This is one of the main advantages of having facilitators and a controller in the system. It improves the flexibility in the integration of agents, which allows the agents to be ignorant or indifferent to other existing agents in their own community as well as in the other communities in the architecture. This eliminates the task of informing the agent or updating its knowledge.

The architecture introduced here is shown in Figure 2. The agents in the same community compete with one

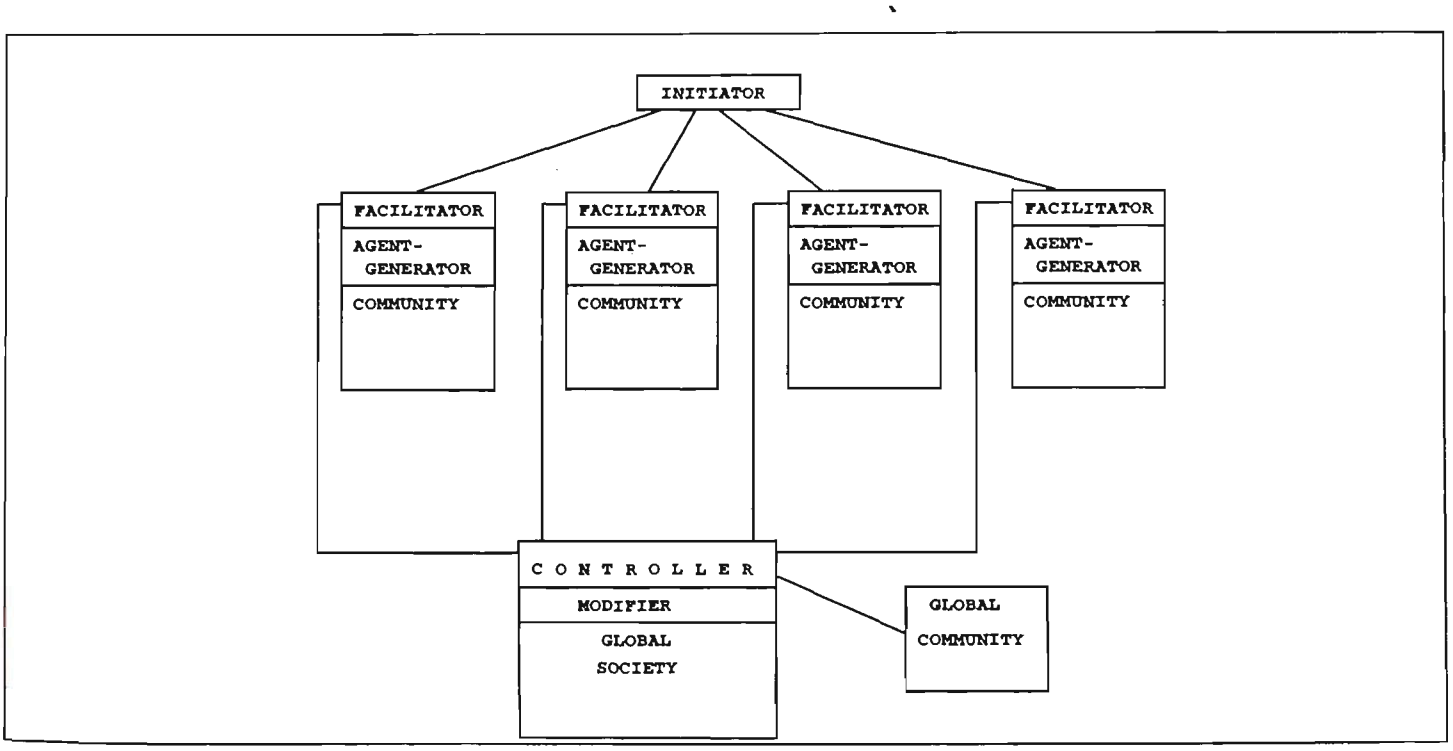


FIGURE 2. System Architecture

another, while agents from differing communities complement one another. The operation of the system begins with the initiator sending a request to the facilitator. The facilitator then selects the agents that qualify the requirements and incorporates them to the

global society managed by the controller. This global society is a base that assembles all the selected agents.

Each facilitator has to transport at least one agent as a representative of its community into the global society. Therefore, if the existing agents' specifications in one community do not match the essential requirements, a new agent has to be created. It is the responsibility of the agent-generator module to generate the new agent with the required specifications.

The controller regulates the interactions among the selected agents based on a collection of constraints and constructs the solution. The collection represents the design constraints and the user's requirements. The solution consists of the coordination of the partial solutions represented by the various agents from the different communities. Given a set of constraints and specification, it is possible to generate more than one solution.

4.1 The Advantages of the Architecture

One paradigm introduced to overcome the complexity barrier is to build systems of smaller and more manageable components which can communicate and cooperate [3]. There are several advantages to the approach of using a distributed artificial intelligence architecture. Firstly, the smaller components are simpler and more reliable because of reduction in complexity. Secondly, decomposition of the system aids the problem of conceptualisation. It also increases the system modularity, thus making the system easier to manage and to understand.

The CBR framework provides a rich experience-based environment. Problem solving using the CBR methodology does not start from first principles, thus reducing time and effort in the formation of an initial solution. The CBR system is also capable of learning and increases its capacity to reason and solve new problems through the expansion of its case library. Case-based systems also provide a paradigm for interacting with an expert system that is useful to both novices and experts. While the novices are provided with a good training ground to gather more experience and learn more about the domain, the experts can use the system to automate simple decisions to aid them in planning, diagnosing or remembering. Moreover, the performance of the system compares favourably to other approach. When the current situation moves out of the system's range of experience, there are fewer cases retrieved. But degradation of the system will be graceful and temporary only. This is because it 'remembers' the new cases and stores them in the case library for future retrieval, thus improving the performance once again.

The use of agents encourage reusability of problem-solving components. Employing object-oriented techniques enables the system to take advantage of the benefits of object-oriented programming. For example, case (agent) representation can be modelled easily and neatly by encapsulating its attributes and behaviour into a single object. Hence, building the knowledge base and the inference engine into the case becomes simpler. In addition, the object-oriented database offers a natural environment for a case-based approach. In addition, the databases are implemented in such a way that the interface is capable of performing certain operations on its members.

Agents communicate via their local facilitator and the controller, thus reducing the communication flows among the agents and the unforseen conflicts that may arise. This mode of communication also helps to increase the orderly flow of effective communication and interactions of the agents. Having a controller to act as a manager in the global base improves the coordination of all the entities in the system. Management of the system becomes more organised and methodical.

5. IMPLEMENTATION OF CAPP

The design and selection of a protection scheme for various parts of the power system requires not only domain knowledge but experience and skill as well [11]. An expert who is good in analogical reasoning may not be good in remembering. Whenever a problem is encountered, it is quite natural for humans to investigate past problems and try to either adopt or adapt the previous solutions to the current situation. Hence, CBR appears to be the most appropriate technique for the development of the protection system.

Besides capturing and providing a rich resource of expertise and experience in stored cases, CBR methodology also simplifies knowledge acquisition. Problem solving involves searching and retrieving similarities of the current problem in the stored cases. Many successful CBR systems have been developed including the use of cases to resolve disputes, design, planning, diagnosis, legal reasoner eg. the JUDGE system and predictions [12, 13, 14, 15, 16, 17, 18, 19].

The representation of the design cases using object-oriented methodology is not only neatly modelled but the features and behaviour associated with the cases (agents) can be easily described. The agents are grouped together to form communities or case libraries and are stored in an object-oriented database. The development of the system involved numerous discussions and interviews with the protection engineers. The knowledge and the expertise acquired during these discussions form the heart of the

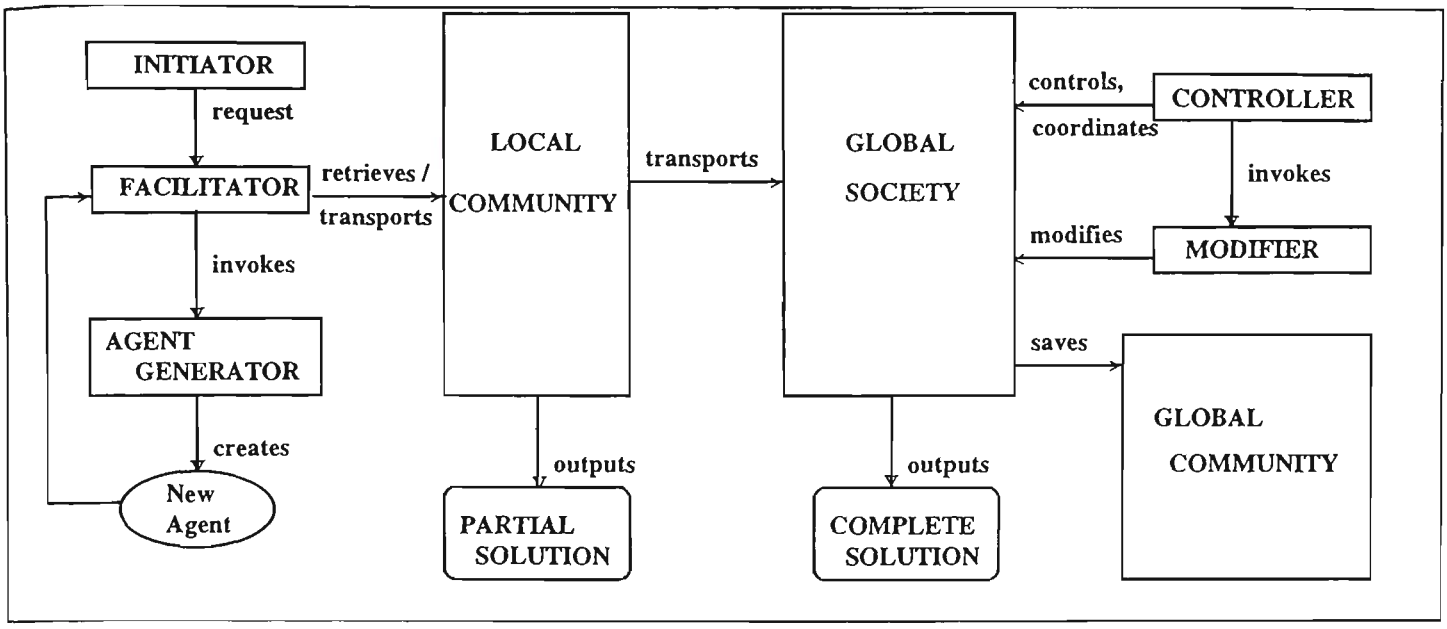


FIGURE 3. Flow of Control

CAPP system and are used to construct the knowledge bases.

The system is being developed in O_2 [21]. O_2 is an object-oriented database management system (OODBS) based on C language. It comes with a complete development environment. At this stage, two components covering the busbar and line protections for a power system have been completed. The agents created are kept as past design cases in the case-library. The flow of control of the system's modules is shown in Figure 3.

Given an initial set of specifications, say, for a busbar, the initiator module invokes the appropriate facilitator. Applying CBR technique involves the facilitator using the set of specifications as a message, and tries to recall the bus agents that match them. These agents are then transported to the global society. If no agent responds to the message which means CBR fails, the facilitator will invoke the agent-generator module. A new bus agent is generated based on the specifications and requirements needed. This new agent, once created has the intelligence to deduce a solution to solve the design problem associated to the busbar. It also has the capability to explain how its solution is derived. Based on some of the new agent's attributes, the facilitator will try for the second time to retrieve any similar agents from the community. This is to ensure that no duplication of agents would exist in one community. However, even though a similar agent is found, the user may disagree with the solution, thereby causing a new agent to be generated.

If the system needs to design a protection scheme which covers several parts of a power system, eg. the bus and lines connected to it, then some coordination is required. This is achieved by interacting the 'retrieved' or new agents in the global society by the controller module. The retrieved cases may have to be modified to adapt the solution to the current problem. Hence, this coordination process may involve invoking the modifier module. The result is a protection design scheme for a power system.

5.1 System Operation

Figure 4 illustrates the system operation and shows part of the system which has been implemented with the exception of the modifier module at this stage.The operation of the system begins with the initiator asking the user some fundamental questions regarding remote protections and current transformers. Examples of the questions asked are whether remote protections are required, whether dedicated current transformers are available and their characteristics. The initiator passes the user specification to the appropriate facilitator. An attempt is then made to retrieve similar cases from the case library. If no suitable cases are found, the agent-generator constructs a new case based on the user specifications. The retrieval process is then repeated to ensure that the newly created case does not exist at all in which the new case will be added to the case library. The retrieved cases or the new case are transported to the global society where the coordination and modification processes may take place, if required.

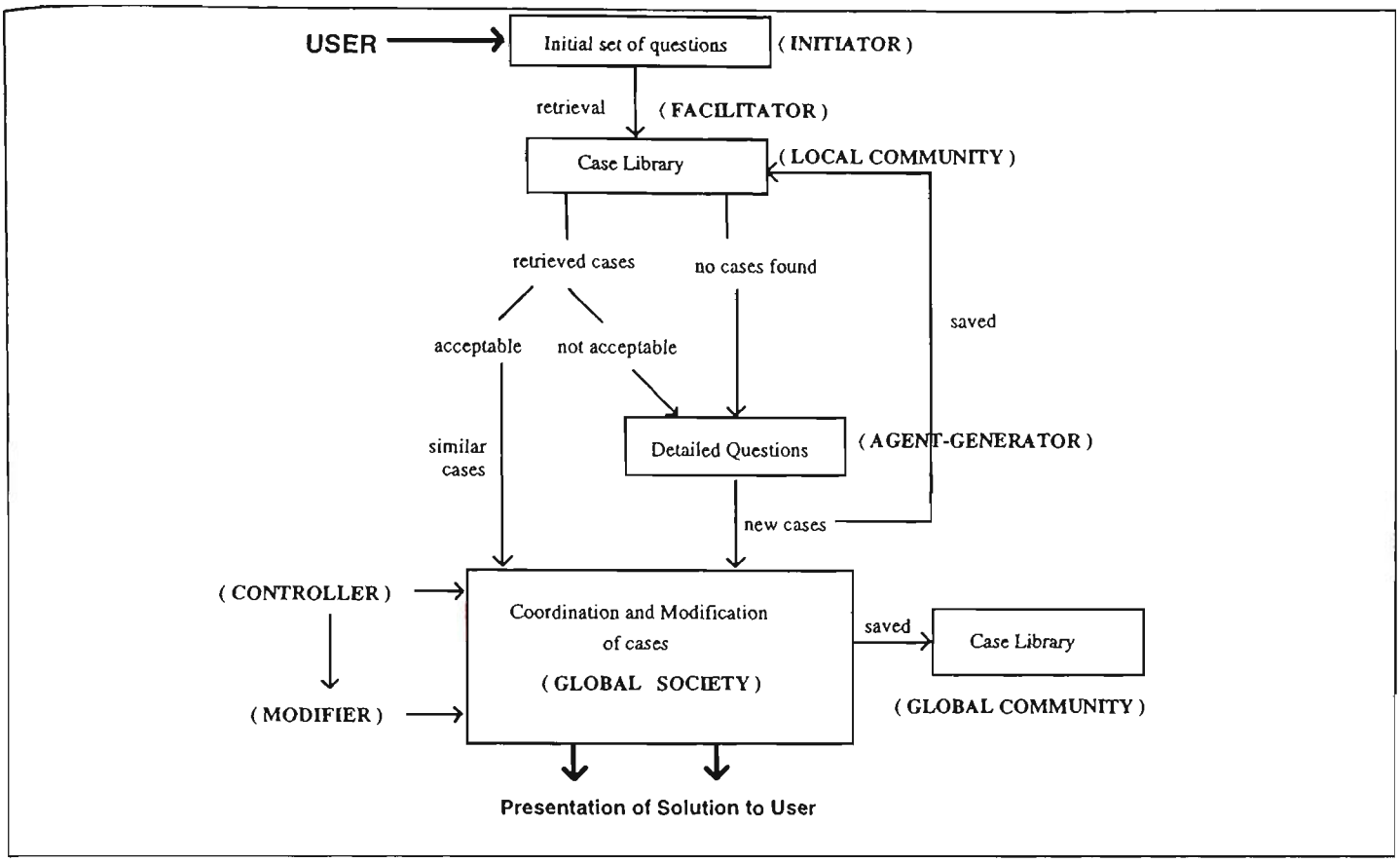


FIGURE 4. System Operation

The result generated by the above procedure represents the solution which is presented to the user and saved into the global community.

6. CONCLUSION AND FUTURE WORK

An architecture of a system for the design, analysis and assessment of power system protection scheme is presented which is being developed using the object-oriented database managment system, O_2. It is based on a CBR paradigm and consists of agents, facilitators, agent-generators, a modifier and a controller. A case-based reasoning approach is particularly appropriate because the expertise is richly captured in the form of past experiences which may be adapted to new situations. Upon given a set of input specifications, appropriate agents are invoked via the relevant facilitators. Coordination of partial solutions are carried out by the controller which is responsible for providing the final design. Although this architecture is principally designed for power system protection, it can be usefully applied to design situations in other applications areas.

One of the main advantages of this architecture is the increased modularity which makes the system more manageable and easier to understand. Moreover, the CBR framework provides a rich experience-based environment which assists and improves the problem solving tasks. At the same time, the employment of facilitators and a controller enhances the management and the coordination of the entities in the system.

The pilot studies and the partial implementation of the system provide a convincing demonstration of the use of CAPP architecture in the area of power system protection. Future work involves the refinement of the explainer component in the agents, the controller and the modifier modules.

7. ACKNOWLEDGEMENT

The authors are grateful to the Australian Electricity Supply Industry Research Board for their funding. The authors would also like to thank Ted Alwast for his invaluable comments and in helping to edit the paper, John Horwood for his suggestions and comments, both from the Computer and Mathematical Sciences Department, Victoria University of Technology and Charles Biasizzo of Power Systems Consulting Pty Ltd. for his advice.

REFERENCES

[UPK93] Uma G, Prasad B.E, Kumari O.N, Distributed intelligent systems: issues, perspectives and approaches, *Knowledge-Based Systems* vol.6 (2) June 1993,pp.77-86.

[Dec87] Decker K. S, Distributed Problem-Solving Techniques: A Survey, *IEEE Transactions of Systems, Man and Cybernectics* vol. SMG-17(5), September/October 1987, pp.729-740

[GK94] Genesereth M. R, Ketchpel S. P, The Software Agents, *Communications of the ACM* vol.37(7), July 1994.

[HG93] Huang G.Q, Brandon J.A, Agents for cooperating expert systems in concurrent engineering design, *AI EDAM* 7(3) 1993, pp.145-158.

[NKS+92] Nishiyama T, Katai O, Sawaragi T, Iwai S, Horiuchi T, A framework for multiagent planning and a method of representing its plan integration, *Transputer/Occam Japan* 4 1992, pp.161-175.

[DDL 92] Devapriya D.S, Descotes-Genon B, Ladet P, Distributed intelligence systems for FMS control using objects modelled with petri nets (scope blackboard), *Manufacturing Systems* 1992, pp.73-77.

[KGT93] Khedro T, Genesereth M.R, Teicholz P.M, Agent-based framework for integrated facility engineering, *Engineering with Computers* vol.9 1993, pp.94-107.

[MCM91] Mac Intosh D.J, Conry S.E, Meyer R.A, Distributed automated reasoning: Issues in coordination, cooperation and performance, *IEEE Transactions on Systems, Man and Cybernetics* vol.21(6) Nov/Dec 1991, pp.1307-1316.

[Bir93] Bird, S.D, Toward a taxanomy of multi-agent systems, *International Journal Man-Machine Studies* vol.39 1993, pp.689-704.

[SS93] Schwartz D.G, Sterling L.S, Using a Prolog meta-programming approach for a blackboard application, *Applied Computing Review* vol.1(1) winter 1993, pp.26-34.

[WKK93] Wong S.K, Kalam A, Klebanowski A, A decision-support system using case-based reasoning in power system protection, *Australasian Universities Power Engineering Conference* 1994, pp.682-687.

[DK93] Domeshek E, Kolodner J, Using the points of large cases, *AI EDAM* vol.7(2) 1993, pp.87-96

[MZ93] Maher M.L, Zhang D.M, CADSYN: A case-based design process model, *AI EDAM* vol.7(2) 1993, pp.97-110

[BZ93] Bardasz T, Zeid I, DEJAVU: Case-based reasoning for mechanical design, *AI EDAM* vol.7(2) 1993, pp.111-124

[Sya88] Syara K, Using case-based reasoning for plan adaptation and repair, *Proceedings of a workshop on case-based reasoning*, Florida, May 10-13 1988, pp.425-234

[Kol88] Kolodner J, Extending problem solver capabilities through case-based inference, *Proceedings of a workshop on case-based reasoning*, Florida, May 10-13 1988, pp.21-30

[Kot88] Koton P, Reasoning about Evidence in Causal Explanations, *Proceedings of a workshop on case-based reasoning, Florida*, May 10-13 1988, pp.260-270

[Lew93] Lewis L, A case-based reasoning approach to the management of faults in communications networks, *IEEE Conference on AI for Applications* 1993, pp.114-120

[Sto94] Stottler R.H, CBR for cost and sales prediction, *AI Expert* August 1994, pp.25-32

[SF93] Shaw M.J, Fox M.S, Distributed artificial intelligence for group decision support, *Decision Support Systems* 9 (1993) pp.349-367.

[Tec93] A Technical Overview of the O_2 System, September 1993.

PROCEEDINGS

1995 International Conference on Energy Management and Power Delivery

EMPD'95

Jointly organised by

National University of Singapore

IEEE Singapore Section, Power Engineering Chapter

SINGAPORE POWER

Singapore Power

Nanyang Technological University

21 - 23 November 1995

The Westin Stamford and Westin Plaza, Singapore

IEEE Catalogue No. 95TH8130

DEVELOPMENT OF A POWER PROTECTION SYSTEM USING AN AGENT BASED ARCHITECTURE

*S.K Wong **A. Kalam

*Department of Computer and Mathematical Sciences
**Department of Electrical and Electronic Engineering
Victoria University of Technology, AUSTRALIA.

Abstract

This paper presents an approach to the development of a system for the design, analysis and assessment of power protection schemes. The system aims to assist the protection experts in automating their work and aids them in diagnosing, planning and remembering. The system could also be used as a teaching or a training tool for the inexperienced fresh graduates in the field. Protection for power system can be viewed as a sum of coordinated protective devices located in the various parts of a power system. The design of protection schemes depends on the configuration of the system, the specifications the system must meet and the constraints that must be satisfied. Selecting and setting the appropriate relays and protective devices are no easy tasks and require skill, experience, heuristics and common sense knowledge. This paper presents a generic architecture based on multi agent paradigm and introduces a novel approach in the development of an intelligent system. It uses distributed problem solving technique and integrates different reasoning methodologies such as case based, rule based and explanation based. The architecture of the system is based on an object oriented paradigm and utilises a multi knowledge representation scheme in a case based framework. Part of the system which has been implemented in an object oriented environment shows a promising and convincing demonstration of the system's architecture and approach.

1.0 INTRODUCTION

The application of expert or intelligent system to power system has grown to become an area of strong research interest in the past few years. However, the total number of research in the area of intelligent power system protection is still low compared to the total number of research carried out in the area of power system. Most of the research work in the protection area are mainly concerned with the development of expert system applications in protective relaying, substation control and related monitoring functions. Examples of expert system applications in the protective relaying field include relay setting and coordination function, selection and coordination of fuses in an industrial customer environment and substation control [Gora and Kacejko 89, Hatta et. al 88, Kawahara et.al 93, Kezunovic et. al 91, Lai 89, Lee et. al 89]. Other areas of research include fault diagnosis, control, monitoring, on-line restoration, maintenance, and tutorial and consultation system [Booth et. al 93, Chen et. al 94, Fauquembergue et. al 93, Lee et. al 94, Marin and Banerjee 94]. However, there are very few applications developed that deals with the design of a protection scheme for a power system.

Protection for a power system can be viewed as integration of the protection for all components of a power system including busbars, motors, generators, transformers, lines. The protection schemes available to a power system varies depending on a number of factors viz. location and importance of the power system, voltage level, components to be protected, economic constraints, availability of resources, utility policies and so on. One major influencing factor is the availability of funds for the implementation of a protection system.

Design has never been an easy job; it is a skill central to many human tasks and in many professions. It is a complicated activity which is always tied to some constraints and relies not only on knowledge of design styles and domain knowledge such as principles and performance theorectical guidance, but also frequent references to previous designs. If a design problem is ill-structured, then design experiences and heuristics play a very important role in the design process. Most designers rely on prior designs and draw upon their knowledge to generate new designs and solve current problems.

There are many expert system applications but not that many decision support systems developed in power system protection. Decision support systems play a more supporting role in aiding the user in decision-making process in selecting the most appropriate decision or course of action. It may also offer or advise the user a number of alternative solutions or the best solution given the problem constraints. On request from the user, the system may advise, examine and evaluate the different options available to the decision maker. Hence, decision-support systems are most successful in application domains where decisions have a more subjective nature and are based on heuristics and experience rather than well defined algorithms.

This paper is concerned with the development of an Intelligent System for Power Protection (ISPP) that aids the engineers in the design of a protection scheme for a power system or a part thereof. One special feature about ISPP is that it is built to imitate the human approach to problem solving, that is, relating to past experiences to solve the current problem. In practise, designers always refer to past experiences and even the work of other designers when they are working on a design problem. ISPP is developed and built based on a generic

IEEE Catalogue No. 95TH8130
0-7803-2981-3/95/$4.00©1995 IEEE

architecture. It is an agent based system using message passing and blackboard as the methods for communication among the agents. The system utilises multi paradigm reasoning strategy which employs various reasoning techniques and applies them at appropriate times to reason a solution or an explanation to the current problem.

2.0 OVERVIEW OF DEVELOPED SYSTEMS

There has been an increase in the number of developed application systems in the area of protection over the recent years. However, many of these systems are mainly on the study and analysis of the power system eg. faults diagnosis, location and calculation, monitoring, protective relaying and alarm processing.

Some of the expert system applications in the protective relaying field deals with relaying and relay setting coordination function [Gora and Kacejko 89, Kawahara et. al 93, Lai 89, Lee et. al 89]. Other programs developed include the calculation of impedances and determination of settings for zones 2 and 3, a high speed digital distance protection scheme and analysis of transmission line insulation for lightning performance [Leung 91, Li et. al 91, Thum and Liew 91].

Knowledge based systems are becoming more popular nowadays. Kalam and Negnevitsky [Kalam and Negnevitsky 93] proposed a logical knowledge base approach utilising expert system techniques as an operational aid for overload clearance, while Marin and Banerjee [Marin and Banerjee 94] developed a knowledge based diagnostic system for the maintenance of transformers and circuit breakers.

Literature survey shows there have been very few applications that focus on aiding the protection engineers in the aspect of designing protection systems. Research in this area has somehow been overlooked even though the applications would be very useful and practical in aiding the engineers in the tedious process of design and assessment of protection systems. Such applications may include calculation of relays settings, and display of relay information. Among the very few applications in this specific area are [Kalam et. al 91, Liu et. al 94, Wong et. al 94].

[Kalam et. al 91] is a knowledge based system developed using Personal Consultant Plus, which plays an advisory role and provides necessary protection scheme for a given power transformer setup. [Liu et. al 94] is an expert system built on OPS83 which plays a similar role in power transformers as well. CAPE is another research project [Enns et. al 92] developed under the sponsorship of ten major U.S electric utilities. It is a productivity tool and a computer-aided system that provides comprehensive computing and record-keeping environment.

The system, ISPP, discussed in this paper is intended to assist engineers in the design and selection of protection schemes for a power system. In addition to that, it could be used as a training or learning tool for graduates and inexperienced engineers in the protection area. ISPP is an intelligent system which could perform several functions such as designing a protection schemes for a given power system or a part thereof, providing explanation as to how a solution is derived, verifying a protection system and providing information of various relays from different manufacturers.

3.0 MULTI PARADIGM APPROACH

ISPP employs a variety of different methodologies and reasoning techniques in an attempt to maximise the benefits and advantages offered by the various methodologies. The system architecture is based on distributed expert systems, also known as agents which operate in an object oriented environment. It employs a multi paradigm reasoning strategy to increase its power and capacity to reason and provide a satisfactory solution or explanation to a problem. These reasoning strategies include case based, explanation based, rule based and argumentation (reasoning under uncertainties) [Huang et. al 94].

Distributed problem solving is a sub-area of AI that concerns with distributing control and data to achieve cooperation, coordination and collaboration among the problem solvers. Case based reasoning is employed as the main reasoner of the system which builds a library of past experiences (cases). It utilises specific knowledge of previously experienced concrete problem situations to solve the current problem. The distributed problem solving approach increases the system modularity, making the system easier to manage and understand. The case based framework provides a rich experience-based environment where problem solving does not have to start from first principle, thus reducing time, cost and effort in the formation of a solution.

3.1 Distributed approach and object orientation

Agent architecture are best suitable for applications where *interoperability* between application programs are required. *Interoperate* means the exchange of information and services with other programs to solve problems that could not be solved alone [Genesereth and Ketchpel 94]. Agent architecture are thus suitable for applications where task decomposition and coordination of solutions are necessary. Hence, communication among agents is inevitable and is important in order to coordinate a final solution. Communication can be achieved via message passing and/or broadcasting using a blackboard structure.

Agent-based software engineering is often compared to object-oriented programming [Genesereth and Ketchpel 94]. Both approaches provide a message-based interface independent of its internal data structures and algorithms but with the primary difference in the interface language. The concepts are compatible, thus the creation of software agents can be achieved more easily using object oriented programming. Moreover, everything in this world can be viewed as an object, eg. a person, a house, a design, etc. An object or in this example, a design work can be related to a case. For example, consider a designer who keeps all designs separately in different folders. Each folder contains one design with all related information and specifications. Here, a folder corresponds to a case. Based on this viewpoint, the object oriented paradigm would then provide the most suitable platform or environment for the development of ISPP. In addition to the inheritance, polymorphism and

encapsulation properties, reusability of software components has made the modelling and the implementation of the system much easier to achieve. All the attributes, specifications, behaviour and properties of an object can be efficiently encapsulated within the object. Even the knowledge base or the domain knowledge can also be neatly packaged within the object.

3.2 Case based reasoning

One of the main reasoner employed by ISPP is case based reasoning. Case based reasoning is chosen because it fits into the application domain naturally. Case based reasoning is a technique that builds a case library with new cases and problem solving is achieved by retrieving similar cases and adapting the solutions to the problem case. The representation of an encapsulated object enables easy storage and retrieval of cases. Furthermore, in practical situation, solving a design work never or very rarely begins from first principle or from scratch. People always refer to their experiences in dealing with similar cases before they start to work on any new problems. This is the general human approach to problem solving. And the fact that case based technique resembles very closely to the natural way of reasoning that people employ in their daily decision making process makes it even more appealing and appropriate to use.

3.3 Explanation based reasoning

Explanation based reasoning is modeled after an explanation based decision making process through which people are believed to perform some decision making tasks [Hair et. al 92]. Alternative explanations can account for a given set of data and the eventual decision taken depends on the strength of each explanation. An explanation is a causal model which incorporates all available data into a coherent structure that supports one of the possible decision outcomes. Hence, explanation based reasoning would be suitable in situations where explanation is required for a cause of action or decision taken for a given set of data. Explanation based is employed in the application system as the explainer to give the user a satisfactory explanation as to why a certain protection scheme is proposed for a particular power system.

3.4 Rule based reasoning

Rule based reasoning is one of the simplest and most common reasoning technique that has been in used in many systems since the early days of expert systems. It takes the form of '*if-then*' construction which is a straight forward reasoning technique and is mainly employed in the development of expert system applications. Even though it has its limits, rule based reasoning is still very popular because it is easy to use and implement. Rules are most efficient and can be just as powerful in systems where the problem domain is well-defined and deduction can be done using forward or backward chaining. ISPP employs rules as the secondary reasoner, that is, when case based reasoning fails to retrieve any similar cases from the database, rules are used to derive a solution to the user's problem.

3.5 Argumentation

In addition to rules being a secondary reasoner, argumentation is also used. Argumentation is a decision procedure based on a simple flexible method of reasoning under uncertainties for argument generation and aggregation [Huang et. al 94]. Decision making is often complicated by the presence of incomplete or even conflicting information. For example, there could be a number of protection schemes that could be applied to a power system. Let's say for a given component of a power system, high impedance differential scheme or medium impedance differential scheme could be applied. Alternatively, overcurrent scheme could also be used. But if high impedance differential scheme were to be applied, dedicated current transformers (cts) must be made available. Other requirements are the cts must be sensitive enough to saturate under all through fault conditions; all cts must be of the same turns ratio and of low reactance. On the other hand, medium impedance differential scheme, which is just as efficient as high impedance differential scheme could be too expensive. But this scheme is not too demanding on the cts characteristics and dedicated cts is not a necessity. At the same time, various ratios of the main cts and their other load than that of the differential scheme are acceptable. Both schemes are unit schemes where backup protection for the downstream component is not possible. The overcurrent protection scheme would be less efficient, ie. slow in operation but it would cover all types of fault on the busbar and backup protection could be arranged. To facilitate decision making in such a context, a domain independent decision procedure must be abstracted and constructed [Huang et. al 94]. This procedure is separated from the domain specific knowledge which also permits the formalisation of decision knowledge.

There is always a goal to be achieved in the process of decision making, represented as a *decision context*, which could be specified by the user or generated by the system in operation. Referring to the example in the previous paragraph, the goal could be to design a protection scheme which has to be fast and efficient and the incurred cost must be taken into consideration. Provision for backup protection is not important. In this context, there are pros and cons for each proposed option. This eventually would be combined to give a most preferred decision - ie. to employ high impedance differential scheme (assuming that dedicated cts are available.)

4.0 THE ARCHITECTURE

ISPP architecture is based on the employment of the methodologies and techniques discussed in the previous section. The main purpose of ISPP is to provide a protection scheme or a range of applicable alternative schemes as a solution to a given power system. A protective scheme consists of a set of protective gears with appropriate settings to trip the relay in order to open the circuit breakers whenever a fault occurs. The protective schemes built for a power system depend on the configuration of the system, the specifications the system must meet and the constraints that must be satisfied.

The generic architecture on which ISPP system is built is shown in Figure 1. Data can be fed into ISPP from either interacting with the user directly or with another independent system.

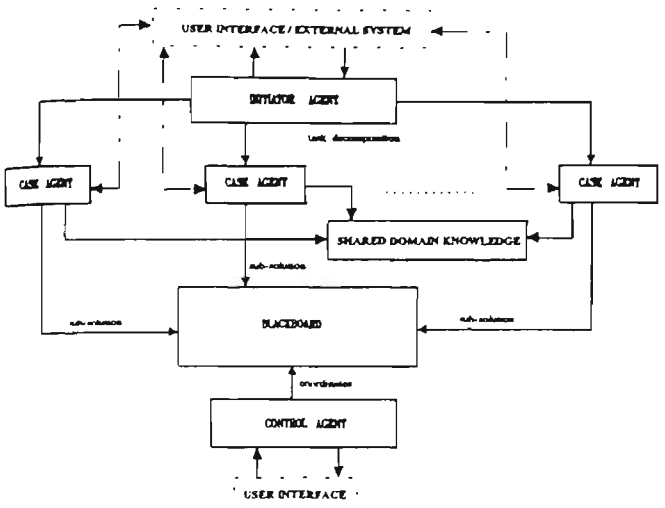


Figure 1. System Architecture

The Initiator Agent is the interface agent that communicates with the external system or the user. It keeps the profiles of all the Case Agents and communication is achieved through message passing. This agent performs task decomposition, that is, it breaks down the problem into smaller tasks and distributes them to the appropriate Case Agents. All data and information received from an external system goes through this agent before the correct information are transmitted to the appropriate Case Agents. If the source of data comes from the user, the Initiator Agent will assess the user's problem and then direct the user to interact with the correct Case Agent.

The Case Agent represents the different components of a power system that need to be protected. Each Case Agent works independently; it employs case based and rule based reasonings techniques - retrieving similar cases from its case library or constructing a new case from scratch, if there are no cases retrieved. However, it could also be instructed to bypass the retrieval process and start the problem solving task from first principles. Either way, a case would be generated which defines the problem and contains the solution for it. Eventually, the cases representing the solutions constructed by the Case Agents involved in the problem solving are posted to the blackboard for coordination by the Control Agent.

The blackboard is a global memory area where information regarding the problem is recorded by the Initiator Agent. Updates are done by the Case Agents as they complete their tasks. In addition to this, the blackboard has a large storage area to contain all constructed cases posted by the respective Case Agents.

The Control Agent serves as the system controller and coordinator. Coordination in this context means coordinating and integrating the various cases to form a single coherent solution. This agent employs multi paradigm reasoning strategies consisting of explanations, rules and argumentation. The coordination process may involve refinements or modifications and adaptations of some of the cases. If the modification process fails because one or a few of the cases are too costly or complicated to modify, the Control Agent would send the case(s) back to the Case Agent(s) concerned, inform them the cause of failure and request for the case(s) to be rebuilt. In such situations, the respective Case Agent would repeat its process again and construct a new case.

4.1 System Development

ISPP is initially developed in O_2. O_2 is an object oriented database management system based on C language. It comes with a complete development environment which provides the object oriented platform for the implementation of ISPP system. At this stage, two components covering the busbar and line protections has been completed. The implementation carried out thus far has provided substantial support for ISPP system as a working model.

4.1.1 The classes

The agents are the software components represented as classes. The functions of the Initiator Agent is to initiate problem solving in the system by decomposing the original problem and distributing the tasks to the various Case Agents.

The problem domain ie. protection for power system can be divided into smaller domain, each focusing on a particular part of the power system, eg. busbars, lines, transformers, generators, motors. Each Case Agent defined in the system represents one part of the power system. Therefore, a Case Agent has its own domain knowledge and inference engine which specialises on solving problems in a particular part of the problem domain. Examples of the Case Agents in the system are *Busbar* and *Lines*. All the components represented by the Case Agents have certain similar attributes. Thus, in order to avoid duplicating codes and to reduce the programming effort, a superclass is created where all the Case Agents classes can *inherit* from it. The functions of the Case Agent are to build a solution to its given problem by applying and adapting the solutions of similar cases from its case library or construct a new case from scratch. The case library which stores past cases is viewed as the agent long term memory.

The Control Agent is basically responsible for coordinating the various solutions into one single integrated solution and providing an explanation as to how the solution is derived, if required. The multi paradigm reasoning strategies are encapsulated within the Control Agent and are represented as the class methods.

The case library and the blackboard are also built as classes. The case library represents part of the system's distributed knowledge and forms the Case Agent *expanding* experiences in solving domain problems. The blackboard is a global memory area to facilitate problem solving, recording information and serves as a work area for the Control Agent.

Object oriented processing provides support for data abstraction, knowledge encapsulation, inheritance, reusability, extensibility and modularity. Thus, applying

object oriented paradigm has reduced the implementation effort and increased the efficiency of the program at the same time.

4.2 ISPP - The Application System

ISPP is designed to communicate interactively with the user. The implementation thus far includes the busbar and line components. Information required by the system include configurations and constraints of a busbar or a line eg. information on the cts, requirements of a protection system, and specifications of the component, if necessary. The system would first automatically attempt to build a solution by applying its experience in dealing with similar problems to solve the current problem. However, if the current problem is new to the system, it starts to build a new case from scratch. Coordination is only required if the protection system to be designed covers more than one component. Figure 2 shows the operation of ISPP and the interactions of the various agents.

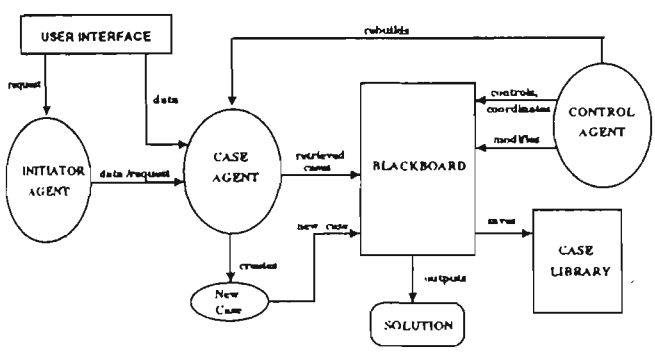


Figure 2. System Operation

Figure 3 presents a flowchart of the system functionality which shows the formation of a single solution case.

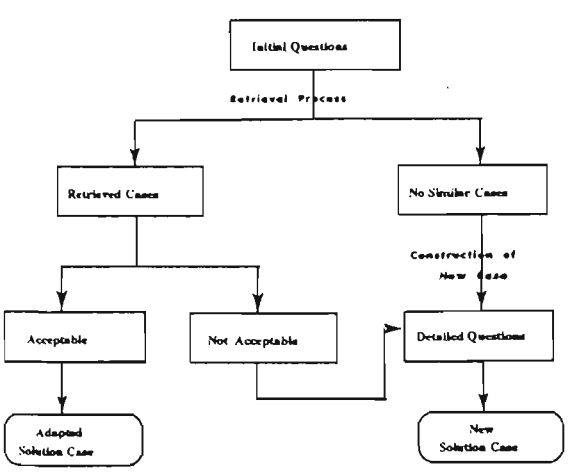


Figure 3. The flowchart of the system operation

The system operation starts with the Initiator Agent identifying, decomposing the user problem and invoking the appropriate Case Agent. The Case Agent would search its memory for similar experiences in solving the current task. If similar cases are found, they are presented to the user for confirmation of acceptance. This step could be automated. If accepted, the solutions of the retrieved cases are adapted to the problem case. However, if the retrieved cases are not acceptable or there were no similar cases found, a new solution case is then constructed. This requires further informations about the problem case which is acquired through interacting with the user. The cases produced by the invoked Case Agents are posted to the blackboard for coordination by the Control Agent. One or some of the cases may need to be modified or rebuilt before the final coordinated solution is presented to the user.

5.0 EVALUATION

Discussions on the approach outlined in this paper have been carried out with a number of protection engineers. The prototype developed in O_2 have been critically examined and evaluated and are generally satisfying. However, it is still undergoing refinements and more tests have to be carried out before the prototype is fully implemented. In view of the promising results obtained so far, and the verification of the approach by the working model in O_2, the system is transferred to DOS environment. In addition to being an applicable and practical system, the DOS version would be more feasible and easier to implement than in a Unix platform. The system is being built in C^{++} language because of its portability. This system would be more user friendly with a graphical interface for line diagrams to be drawn, modified and retrieved and requires less computing knowledge on the part of the engineers.

6.0 CONCLUSION

The approach to the development of a practical system for the design and assessment of power system protection schemes has been presented. The system consists of several distributed expert systems, known as agents, as opposed to having just one expert system in the traditional systems, and are organised in a hierarchical structure. Each agent is an expert in solving problems in its own specialised domain. The knowledge base is distributed and represented in different forms. For example, the Case Agent has knowledge in the form of experiences (case library) as well as in the form of rules and facts in its domain knowledge which is utilised to deduce solutions from scratch. Multi paradigm reasonings strategy is employed to enrich the system's reasoning concepts and increase its reasoning capacity. Different reasoning techniques are applied at different times depending on the situation. A blackboard architecture is incorporated into the system to facilitate problem solving.

The adopted approach offers several advantages. Among them are the distributed knowledge base in multiple forms enhances the problem solving capability by integrating the domain knowledge with past experiences. It also enables more complex problems to be solved. Moreover, the distributed application encourages modularity and increases the speed of problem solving by decomposing a problem and distributing the tasks to the problem solving agents instead of having just one expert system to perform all the various tasks in the entire system.

The application of case based reasoning technique makes the Case Agents more competent, useful and intelligent through the dynamic growth of its experiences in problem solving. Furthermore, case based technique is known to handle incomplete information or missing data quite well, that is, the features of the missing values is not used in the retrieval process.

Other advantages offered by distributing problem solving approach and object oriented paradigm include reliability, reusability, robustness (the system can still operates even if one part of the system may fails), reduction in the complexity of control and execution of problem task, etc.

Future work involves further development on the Control Agent and refining its functionalities on the working model. The implementation of the DOS version would be carried out incrementally at the same time based on the working model.

References

[APSCOM 91] International Conference on Advances in Power System Control, Operation and Management, 1991, Hong Kong.

[Booth et. al 93] Booth C., McDonald J.R., Stewart R.W., Laycock W.J., Bennett A., Coordinated Control and Protection: Functionality Enhancements via Information Technology, *DPSP 93*, pp.25-28.

[Chen et. al 94] Chen J., Wang P., Fu S., Wang M., Yu E., Development of an Object-Oriented Expert System for Power System On-Line Restoration after a Blackout, *ICPST 94*, pp.1061-1065

[Enns et. al 92] Enns M.K., McGuire P.F., Ramaswami R., Arbor A., CAPE: The Computer-Aided Protection Engineering System, *Proceedings of the American Power Conference*, 13-15 Mac 92, vol.2, pp.1084-1089.

[Fauquemberque et. al 93] Fauquembergue P., Maizener A., Parant J.M., Perrot L., Bertrand H., Monitoring of Protection System Behaviour Using An Expert System Which Analyses Substations Sequential Events Recording Field Experience At Electricite De France, *DPSP 93*, pp.42-45.

[Gora and Kacejko 89] Gora S., Kacejko P., Application of expert system for determination of protective devices' settings, *Proceedings 2nd Symposium Expert System Applications to Power Systems (ESAPS)*, Seattle, WA., USA, 1989, Electric Power Research Institute, Palo Alto, CA., pp 251-256.

[Genesereth and Ketchpel 94] Genesereth M.R, Ketchpel S.P, Software Agents, *Communications of the ACM* vol.37 (7) July 1994, pp.49-54.

[Huang et. al 94] Huang J, Jennings N.R, Fox J, An Agent Architecture for Distributed Medical Care, London, Technical Report 1994.

[Hair et. al 92] Hair D.C., Pickslay K., Chow S., Explanation-Based Decision Support In Real Time Situations, *Proceedings of the 1992 IEEE International Conference on Tools with AI*, Arlington, VA, Nov 1992, pp.22-25.

[Hatta et. al 88] Hatta M., Satoh H., Seriwaza Y., Application of Expert System to the Determination of the Operational Coordination of Switch Gears and Power Fuses for Short Circuit Protection, *Proceedings First Symposium Expert System Applications to Power Systems (ESAPS)*, Research Institute Technology, Stockholm, Sweden, 1988, pp.6.27-6.34

[Kalam et. al 91] Kalam A., Klebanowski A., Poloni D., Knowledge Based System for Transformer Protection, *Proceedings on IEE International Conference on Advances in Power System Control, Operation and Management 1991*, pp.443-448.

[Kalam and Negnevitsky 93] Kalam A., Negnevitsky M., Knowledge Based Approach To Clearing Overloads On The Power System Plant, *Proceedings of Second International Conference on Modelling and Simulation*, Victoria University of Technology, Melbourne, Australia, July 12-14 93, pp.355-363.

[Kawahara et. al 93] Kawahara K., Sasaki H., Kubokawa J., Sugihara H., Kitagawa M., An Expert System for Supporting Protective Relay Setting for Transmission Lines, *DPSP 93*, pp.203-206

[Kezunovic et. al 91] Kezunovic M., Watson K., Russell B.D., Heller P., Aucoin M., Expert System Applications to Protection, Substation Control and Related Monitoring Functions, *Electric Power Research*, 21, 1991, pp.71-86

[Lai 89] Lai L.L., Development of an Expert System for Power System Protection Coordination, *Fourth International Conference in Power System Protection*, 11-13 April 1989, pp. 310-314.

[Leung 91] Leung K.W., Computer-Aided Setting Calculation for Distance Zone 2 and Zone 3 Protection, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.152-157

[Li et. al 91] Li K.K, Cheung C., Xia Y.Q., High Speed Digital Distance Protection -Real Time Simulation and Hardware Development, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.95-100.

[Liu et. al 94] Liu L., Gao Z., Yang Q., Tong W., Construction of Expert System for Designing Protection of Power Transformer, *ICPST 94*, pp.1380-1384.

[Lee et. al 94] Lee H.J., Park Y.M., Hyun S.H., Chu J.B., Yoon Y.B., Development of an Intelligent Support System at Local Power Control Center in Korea, *ICPST 94*, Beijing, China, pp.494-497.

[Lee et. al 89] Lee S.J., Yoon S.H., Moon M.C., Yang J.K., An Expert System for Protective Relay setting of transmission systems, *Proceedings PICA Conference*, Seattle, WA., USA, 1989, IEEE, New York, pp. 296-302.

[MS 93] Second International Conference on Modelling and Simulation, Melbourne, 1993.

[Marin and Banerjee 94] Marin M.A., Banerjee S.N., An Approach to a Knowledge-Based Diagnostic System for Power Equipment Maintenance, *ICPST 94*, pp.717-720.

[Thum and Liew 91] Thum P.C., Liew A.C., Computer Analysis of Transmission Line Insulation for Lightning Performance, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.780-790.

[Wong et. al 94] Wong S.K., Kalam A., Klebanowski A., A Decision Support System Using Case-Based Reasoning in Power System Protection, *AUPEC 94*, pp.682-687.

ISAP'96

ANDO - FLORIDA - USA - JANUARY 28 - FEBRUARY 2, 1996

PROCEEDINGS
F THE INTERNATIONAL CONFERENCE ON
INTELLIGENT SYSTEMS
APPLICATIONS
TO
POWER SYSTEMS

Editors: O. A. Mohammed & K. Tomsovic

SPONSORED BY:
IEEE POWER ENGINEERING SOCIETY
IEEE NEUTRAL NETWORK COUNCIL
THE NATIONAL SCIENCE FOUNDATION

ORGANIZERS:
FLORIDA INTERNATIONAL UNIVERSITY
WASHINGTON STATE UNIVERSITY
THE UNIVERSITY OF WASHINGTON

WITH THE PARTICIPATION OF:
IEEE MIAMI SECTION
THE FLORIDA COUNCIL OF IEEE

Distributed Intelligent Power System Protection Using Case Based and Object Oriented Paradigms

*S.K Wong , **A. Kalam
*Department of Computer and Mathematical Sciences
**Department of Electrical and Electronic Engineering
Victoria University of Technology, AUSTRALIA.

Abstract - **Design of power system protection depends on the configuration of the system, the specifications the system must meet and the constraints that must be satisfied. This paper presents a novel approach in the development of an intelligent system in the protection area of power system. The architecture of the system is generic and is based on multi agent paradigm in an object oriented environment and utilises a multi knowledge representation scheme in a case based framework. A case represents a protection design scheme for a particular component of a power system. In this paper, busbar and line protection is considered under typical constraints and specifications. Agents can be regarded as intelligent entities which specialise in specific components of the power system and are responsible for generating partial solutions. The final solution to the design of a protection system is obtained by coordinating the partial solutions.**

1.0 Introduction

The expert system paradigm has increasingly been used in the development of power system applications. The areas of interest include power system management, system restoration, forecasting, protection, control, distribution, transmission, communication, scheduling, monitoring, security and many more. However, the number of research projects in intelligent power system protection is relatively small. These include application and simulation systems developed for setting calculations for distance protection in zones 2 and 3, load forecasting, distance relaying, alarm processing, fault analysis and transformer protection [1, 5, 8, 10, 11].

However, it is worth noting that: (i) most of the applications are expert system applications which basically employ production rules with backward and/or forward chaining processes; (ii) applications concentrate mainly on the analysis of power systems eg. alarm processing, fault calculations and relaying. It follows that comparatively few knowledge based or decision support systems have been developed and very few applications actually involve the design and analysis of the performance of protection systems.

Expert systems have been successfully applied to well defined problems. Major difficulties arise however, when dealing with problems which are ill defined and, at the same time, are highly diversified. The difficulties are due to [2]:

- Building and maintaining the consistency of a complex rule based system;
- Integrating various knowledge representations and different reasoning strategies used;
- Restricting the problem domain of an expert to a specific area.

Some of the expert systems developed in the protection area are discussed in [4, 9]. Most of these systems involve a large number of formal expert rules. To maintain such knowledge bases which may involve frequent or numerous updates would be very difficult if not impossible task.

An example of an application system which uses relations to represent domain knowledge for power system is called CAPE [3]. CAPE is a productivity tool developed on InterBase - a commercial relational database management system - consisting of a large number of tables. However, the process of updating and querying such tables could be slow whilst maintaining consistency and integrity of the database could prove cumbersome. In general, there are small number of applications [7, 10, 15] that focus on assisting engineers in the design and assessment of protection schemes. Both [7, 10] are relatively small systems that focus only on transformer protection and utilise production rules. However, such applications may be expanded to include verification of protection systems, calculation of relay settings and display of relay information.

This paper presents a generic architecture for power system protection based on multi agent paradigm. The architecture consists of a number of distributed expert systems, also known as intelligent agents, embedded in an object oriented environment. The agents utilise a multi knowledge representation scheme within a case based framework and employ a multi paradigm reasoning strategy to improve problem solving. The multi reasoning strategies include case based, rule based, explanation based and argumentation (reasoning under uncertainties). Moreover, the system also utilises a blackboard which forms an integral part of the system. A blackboard is a global database where selected cases for problem solving are recorded. The blackboard is used as a medium for communication and as an integration platform for posting

0-7803-3115-X/96$5.00 © 1996 IEEE.

and coordinating partial solutions. The proposed system called ISPP (Intelligent System for Power Protection) is intended to assist engineers in the design, selection and analysis of protection schemes. It could also be used as a training or learning tool for inexperienced engineers and new graduates in protection area. Systems such as ISPP cannot be used to replace protection engineers but it represents a very practical and useful system in industries where the supply of protection engineers are low or their expertise are not easily available.

2.0 Protection of Power Systems

Protection for power system is basically viewed as integrated and coordinated protection for all parts of power system. There are varieties of protection schemes available and each scheme depends on factors such as location and importance of the power system, the components to be protected, economic constraints, availability of resources, budget allocation, utility policies and implementation cost.

The design of protection scheme for power system is not a simple task. It represents tedious work that requires a great deal of expertise and experience on the part of the engineer. In addition, the decisions regarding protection are at most times subjective and follow 'rules of thumb'. The protection schemes for power system elements must be designed so that the system meets the reliability requirements, speed and selectivity as set by the system operating constraints [15].

The possible implications of an incorrect protection operation or even delayed fault clearance can be disastrous. Such events will progressively multiply and their likely consequences and effects will continuously increase. Fault conditions are usually not simple events and may have a multiplicative effect on other parts of the system. Therefore, it is imperative that proper and adequate protection to all parts of the power system must be provided.

Given the brief account of the responsibilities of a protection engineer and the protection requirements for a power system, the approach proposed in this paper advocates integration of various methodologies when developing ISPP.

3.0 Overview of ISPP

The system's architecture consists of a group of loosely coupled and decentralised problem solving agents. The system is a distributed knowledge based system where agents have to cooperate and combine their knowledge. The agents are organised in a hierarchical structure and employ multi paradigm reasoning strategy to solve their individual tasks. The system incorporates a blackboard as an integration platform to facilitate problem solving ie. the coordination and integration of partial solutions into a single coherent design solution.

The system has three types of agents: Initiator, Control and Case agents. The expertise required to solve the domain problems is partitioned among domain-specific agents. Each agent architecture differs in the structure and organisation of the knowledge. The knowledge base is functionally separated into domain knowledge and inference knowledge. The domain knowledge contains facts and rules about the domain but it provides no information about how the knowledge should be utilised. The inference knowledge [6] with built-in reasoners is required to specify the inference relations. The advantages of this separation include simplicity in the representation and maintenance of the knowledge, reusability is encouraged, acquisition and modifications to any part of the knowledge can be achieved independently without any side effects on the other part of knowledge.

Fig. 1 shows the organisation of the system's agents.

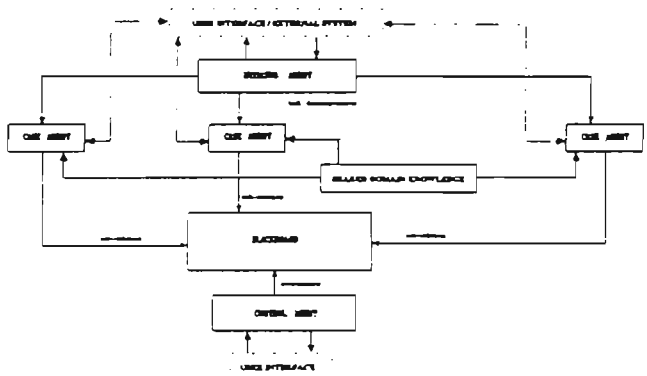


Fig. 1. Organisation of the system's agents

The Initiator agent interacts with the user or an external program. The user may specify the type of problem to be solved or alternatively, an external program could provide the system with the parameters of a power system. Using the input, a problem case is constructed. A problem case represents the specification and features of a component of the power system for which the protection scheme has to be designed. The Initiator agent initiates the problem solving task by decomposing the problem and assigning the various (sub) tasks to the appropriate Case agents. At the same time, the identity of the Case agents are posted onto the blackboard.

The Case agent may represent an expert system in busbars, lines, generators or any part of the power system requiring protection. It attempts to solve a problem by first applying its previous experiences. This is accomplished by retrieving similar cases from the case library. An example is illustrated in section *3.3.1*. The case library forms part of the distributed knowledge which stores historical cases. All new cases that are different from the existing cases in the case library will be automatically stored in the library. When no similar cases are retrieved, problem solving begins from scratch. In addition to the built-in knowledge, the Case agent also has access to a shared domain

knowledge that contains facts about the domain. This knowledge includes related information provided by different manufacturers on the type of relays available in the market place.

The Control agent is responsible for coordinating all the cases that are posted to the blackboard. In coordinating the cases, the Control agent may need to modify or adapt the cases. However, modifications would not be carried out if they have an adverse effect on other cases. In this instance, the case is returned to the originating Case agent for rebuilding. Once a solution is built, the user may request an explanation about a particular scheme.

3.1 Agent Architecture

Object oriented processing provides support for data abstraction, knowledge encapsulation, inheritance, reusability, extensibility and modularity. The message passing capability of object oriented processing allows the system to keep knowledge about the domain separate from the knowledge about reasoning. Using this model, agents can be structured using classes.

Fig. 2 shows the architecture of a Case agent in the system. The domain knowledge represents the agent knowledge and expertise in a specific area of the domain. The inference knowledge is the knowledge applied to task management and decision making. The work area is similar to the human short term memory in which temporary data are registered whereas the control is akin to the human mind. The control is responsible for scheduling the agent activities. It is also responsible for calling the appropriate reasoning strategy (in the inference knowledge) and applies the domain knowledge to solve problems.

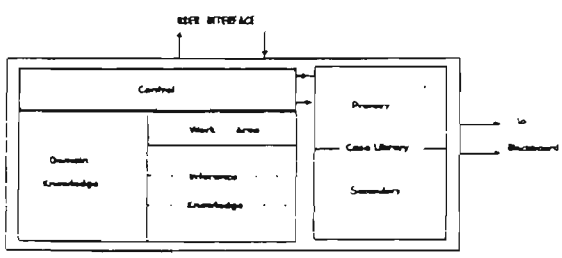


Fig. 2. Case Agent Architecture

The Case agent has an additional feature, which differs from the Initiator and Control agents. It is capable of storing historical cases in its memory. The memory (also known as case library) is similar to the human long term memory and it represents the agent acquired and accumulated experiences. The case library is divided into two memory sections: primary and secondary. The primary stores the latest and frequently retrieved cases, whereas the secondary stores old cases that have not been used for a defined period of time. Retrieval is carried out from the primary memory only. In situations where there is no similar cases found, retrieval is then performed on the secondary memory. This organisation and memory management is meant to provide a more efficient and faster retrieval process.

3.2 Agent Functions

The agents in the system have special responsibilities and each of them has been designed to perform certain functions. To enable the agent to execute its functions more effectively, the inference knowledge of the agent is organised into a number of distinct layers. In fact, each inference layer represents a different reasoning strategy. A detailed discussion of the multi reasoning strategies will be presented in a separate paper.

The Initiator agent mainly employs rules to carry out its functions. Its main responsibilities include problem decomposition, distribution of tasks to the Case agents and writing the identity of the Case agents onto the blackboard. If data is to be input by a human operator, the Initiator agent will direct the operator to interact with the appropriate Case agent.

The Case agent is responsible for generating a solution to a specific task. [1]It uses case based reasoning to retrieve similar cases from the case library and then tries to adapt the solutions. However, if there are too many cases retrieved, it automatically sieves through them in order to reduce the number of potential candidate solutions. As a consequence, a predefined number of cases that best match the current problem will be chosen. This improves the efficiency of problem solving. If no similar cases are found, the Case agent will construct a new case from first principles.

The Control agent also employs multi paradigm reasoning strategy which includes rules, argumentation and explanation based reasoning. Its responsibilities are coordinating and possibly modifying the cases in the blackboard to produce a coherent and integrated solution. The Control agent is also capable of providing an explanation of how a solution has been derived.

3.3 Operation of the System

A prototype has been developed on Unix platform using an object oriented database management system, called O_2. O_2 is based on C language, and it comes with a complete development environment which provides the object oriented platform for implementing ISPP. At this stage, the prototype represents a part of the power system which includes the busbar and the line components only. The following agents have been implemented - Initiator, Busbar, Line and the Control agents.

The type of protection schemes applicable for each component depends on the component's position in the

[1]Fuzzy technique could be applied to select the most appropriate case as a possible candidate since the selection of best matched cases is usually fuzzy and indeterministic.

power system. If the system involves several components, design normally commence with the downstream component and coordination will be required as well.

The flowchart shown in Fig. 3 depicts the system operation. Consider the situation where the user wants to design a protection system for a busbar. Given the information on the current transformers (cts), requirements for back-up protection and so on, the system will search for similar cases. If found, the solutions of the retrieved cases will be applied and adapted to the given busbar configurations and presented as the possible design solutions. If the adapted solution is not acceptable or no similar case is found, the system applies its domain knowledge to derive the design solution from first principles. In this instance, more information with regards to the busbar will be required.

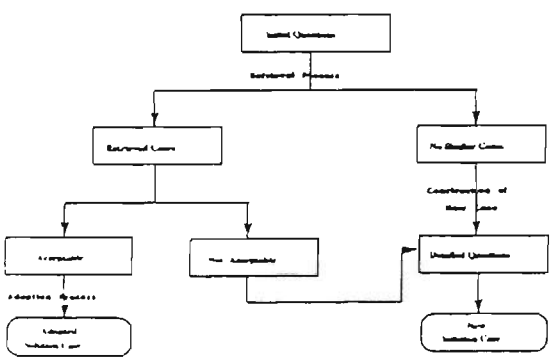


Fig. 3. The flowchart of the system operation

The following section shows an example run of the program, which also illustrates the agents activities and interactions.

3.3.1 Illustration of a Program Run

Let's say the problem is to design a protection system for a bus with a line located upstream as shown in Fig. 4. In this illustration, the bus is required to provide backup protection to an existing system located downstream from the bus, while the line requires a backup protection from an existing system located upstream from it.

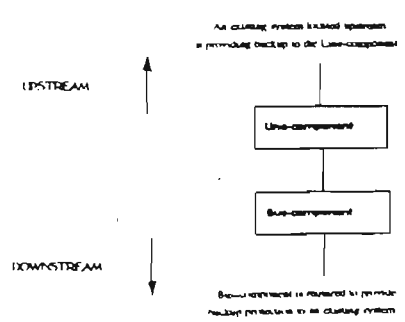


Fig. 4. Components of a section of a power system requiring protection

The program starts with the user specifying the power system or part of a power system that needs protection to the Initiator agent (IA). The IA decomposes the problem and distributes the tasks to the appropriate Case agents, ie. Bus and Line agents (BA and LA) and writes on the blackboard. Upon receipt of the assigned task, the BA/LA will interact with the user for more information regarding the new designs. Typical questions include name of component, remote back-up protection, availability of dedicated cts, cts ratio, likelihood of the cts to saturate, total clearance time required. After the interaction, the BA/LA then commence the design process. Each agent will first attempt to retrieve similar designs from its memory and if successful, will adapt the retrieved cases to the requirements of the current system. Otherwise, the agent will construct the protection scheme from first principles. [2]When the BA/LA completes its assigned task, it then sends the completed case to the blackboard.

When both the BA and LA have completed their designs, the Control agent (CA) will first of all, attempt to coordinate the cases, which may require some modifications. If the modifications are too costly or complicated, the CA will send the case back to the originating Case agent for rebuilding and the design process is repeated.

4.0 Advantages Of The Agent Based Architecture

Due to the nature of the distributed system, each agent contains only partial domain knowledge and hence, the agent can be kept separate and independent. This means changes to the agent architecture can be made independently without affecting other agents. The distributed paradigm allows complex problems to be decomposed into manageable tasks. In addition, the best suited paradigm for solving specific tasks can be encapsulated within an agent. The problem solving capability is further enhanced by integrating the domain knowledge with past experiences, ie. using case based paradigm.

[3]Applying case based reasoning technique enables the Case agents to dynamically increase their knowledge and experiences. This new experiences can then be used to solve a wider range of problems. Furthermore, case based systems can handle incomplete information or missing data quite well [13]; that is, the features of the missing values will not be used in the retrieval process. If the solution or outcome of the retrieved case(s) is acceptable, then the missing value is proved to be unimportant. Otherwise, a new solution can be derived using some other reasoning technique.

---

[2] If no similar designs retrieved, the BA/LA will request more details regarding the system to be designed from the user and build a new design from first principles.

[3] Artificial neural nets could also be possibly used in agents learning.

The decomposition of the problem into domain specific tasks increases the efficiency of problem solving [14]. This approach is far more superior than the conventional approach which involves the construction of a single large expert system which has to perform all the tasks of problem solving. It is even more effective when implemented in a parallel processing environment. The latter type of systems are too monolithic and inflexible, and are usually not sustainable to any changes in the knowledge base.

In general, other merits of using distributed architectures include reduction in the complexity of control and execution of problem task, increased modularity, speed, reliability and reusability. Furthermore, knowledge acquisition becomes simpler, that is, it is easier to find experts in narrow and specialised domain and if the system fails, degradation of the system performance would be graceful (soft crash rather than hard crash).

Other advantages of ISPP architecture is the use of object oriented paradigm which has provided a number of advantages particularly in the areas of modelling and system construction. As stated earlier, the agents can be structured and organised easily and efficiently into classes while their domain and inference knowledge can be encapsulated in each class more naturally.

5.0 Conclusion

This paper presented a multi agent system which integrates case based and object oriented approaches in the development of a distributed intelligent system in the area of power system protection. The approach has been discussed with a number of protection engineers. It appears that case based methodology closely resembles the approach employed by the engineers when designing protection schemes. The prototype implemented so far has been tested and is still undergoing refinement. The protection schemes designed by the system have been critically examined and evaluated by the protection experts. However, more tests still need to be carried out before the prototype is fully implemented.

One of the main advantages of ISPP architecture is the increased modularity due to distributed control and the integrated approach, which makes the system more manageable and easier to maintain. The employment of agents to carry out domain specific tasks enables complex problems to be solved more effectively and efficiently. It also offers more flexibility, integrity, robustness and many other advantages over and above conventional approach which involves only one single large expert system.

The object oriented technique has assisted in the modelling and development of the system. The system can be readily implemented with the mechanisms offered by object orientation such as software reusability, inheritance, encapsulation, modularity and thus can be easily modified or expanded. Furthermore, the employment of multi paradigm reasoning strategies allows the system not only to choose the appropriate strategy but also to switch between the reasoning paradigms, thus mimicking the behaviour of a human expert.

For the future, further refinements to the agents have been proposed and also the introduction of additional Case agents eg. Transformer, Generator and Motor.

Acknowledgment

The authors would like to thank Ted Alwast for his invaluable comments and help in editing the paper, Clement Leung and John Horwood for their comments, all from the Computer and Mathematical Sciences Department, Victoria University of Technology.

References

[1] APSCOM 91, *IEE International Conference on Advances in Power System Control, Operation and Management*, November 1991, Hong Kong, pp.152-157.

[2] Chi T., Kiang Y., Turban E., Distributed Intelligent Agents in Decentralised Organisations, 1993.

[3] Enns M.K., McGuire P.F., Ramaswami R., Arbor A., CAPE: The Computer-Aided Protection Engineering System, *Proceedings of the American Power Conference*, 13-15 Mac 92, vol.2, pp 1084-1089.

[4] Fauquembergue P., Maizener A., Parant J.M., Perrot L., Bertrand H., Monitoring of Protection System Behaviour Using An Expert System Which Analyses Substations Sequential Events Recording Field Experience At Electricite De France, *DPSP 93*, pp 42-45.

[5] Gora S., Kacejko P., Application of expert system for determination of protective devices' settings, *Proceedings 2nd Symposium Expert System Applications to Power Systems (ESAPS)*, Seattle, WA., USA, 1989, Electric Power Research Institute, Palo Alto, CA., pp 251-256.

[6] Huang J, Jennings N.R, Fox J, An Agent Architecture for Distributed Medical Care, London, *Technical Report 1994*.

[7] Kalam A., Klebanowski A., Poloni D., Knowledge Based System for Transformer Protection, *Proceedings on IEE International Conference on Advances in Power System Control, Operation and Management 1991*, pp.443-448.

[8] Kalam A., Negnevitsky M., Knowledge Based Approach To Clearing Overloads On The Power System Plant, *Proceedings of Second International Conference on Modelling and Simulation*, Victoria University of Technology, Melbourne, Australia, July 12-14 93, pp.355-363.

[9] Kawahara K., Sasaki H., Kubokawa J., Sugihara H., Kitagawa M., An Expert System for Supporting Protective Relay Setting for Transmission Lines, *DPSP 93*, pp.203-206.

[10] Liu ,L., Gao Z., Yang Q., Tong W., Construction of Expert System for Designing Protection of Power Transformer, *ICPST 94*, pp.1380-1384.

[11] Lee S.J., Yoon S.H., Moon M.C., Yang J.K., An Expert System for Protective Relay setting of transmission systems, *Proceedings PICA Conference*, Seattle, WA., USA, 1989, IEEE, New York, pp. 296-302.

[12] Marin M.A., Banerjee S.N., An Approach to a Knowledge-Based Diagnostic System for Power Equipment Maintenance, *ICPST 94*, pp.717-720.

[13] Stottler R.H. CBR for cost and sales prediction, *AI Expert* August 1994, pp.25-32.

[14] Smith R.G, Davis R, Frameworks for Cooperation in Distributed Problem Solving, *Transactions on Systems, Man and Cybernetics* vol.SMC-11 (1) January 1981.

[15] Wong S.K., Kalam A., Klebanowski A., A Decision Support System Using Case-Based Reasoning in Power System Protection, *AUPEC 94*, pp.682-687.

Published in

The 31st Universities Power Engineering Conference

International Conference on

Electric Power Engineering,

Technological Educational Institute, Iraklio

Crete, Greece

18-20 September 1996.

Intelligent Power System Protection using Agent Technology

*S.K. Wong **A. Kalam

*Department of Computer and Mathematical Sciences,
**Department of Electrical and Electronic Engineering,
Victoria University of Technology, AUSTRALIA.

Abstract
Agent technology has been applied widely to many diversified areas including medical, industrial control, networking, engineering design and power system such as electricity distribution and supply, and electricity transmission and distribution. However, there has not been any agent based system in power system protection. This paper introduces the architecture of an intelligent multiagent system for the design of protection schemes for power system. The system uses multiparadigm approach and integrates artificial intelligence with distributed problem solving technique. The system consists of three types of agents, namely Interface agent, Coordinatorl agent and Design agent which communicates using message passing paradigm. The agent is designed using the human mind as an analogy which consists of a conscious nund, a logical mind and a memory that are organised in layers. The system knowledge is distributed among the different agents. It is the ability of the agent to use multiple strategies while switching between them whenever necessary, that tends to reflect the human expert's way of thinking. This appears central to the success of the system. A successful prototype system has been built which is detailed further in this paper.

1.0 INTRODUCTION

There have been many applications developed in power system over the past years in the area of fault location, alarm processing, energy management system, electricity transmission, distribution and many more [1, 2, 3]. However, most of the systems developed in the past are mainly expert systems which use rules with forward or backward chaining as the inference engine. Lately, there have been a great deal of interests generated in agent technology. It is very difficult to define what is an agent as there is no single universally accepted definition for it. However, according to Bussmann and Muller [4], an agent can be regarded as an enclosed system which has an interface through which it can interact with its environment. In other words, an agent is like a black box where its interaction is visible to everyone but its realisation may be unknown. An agent could be a human being, a robot, a procedure, or anything that could be treated separately. Among the few agent based systems developed successfully include electricity distribution and supply, electricity transmission and distribution, electricity distribution and supply network, and electricity transportation management and particle accelerator control [5, 6].

It is noted that there is a lack of intelligent systems in the area of power system protection that deals with the design of protection schemes for a power system [7, 8, 9]. It could probably because these type of systems are not real-time systems and majority of researchers and engineers are more interested in developing real-time systems. Both systems developed by Liu *et al.* and Kalam *et al.* are mainly expert systems which are concerned with transformer protection only. Moreover, expert systems are monolithic systems that are more successful if applied to well defined problems. Major difficulties arise however, when dealing with problems which are ill defined and, at the same time, are highly diversified.

This paper discusses the architecture of an intelligent multiagent system and the development of such a system in the area of power system protection. To develop such a system, an expert system would not be sufficient due to its limitations and inflexibility. However, an intelligent multiagent based system which uses artificial intelligence and distributed problem solving techniques, and multireasoning paradigms which may include case based, argumentation (reasoning under uncertainties), explanation based and rules, could be much more appropriate. The reason being that distributed problem solving technique helps to achieve results in the presence of complexity and uncertainty through cooperation [10]. In addition to that, multiagent systems are best suitable for applications where *interoperability* between application programs is required. *Interoperability* is the ability to exchange information and services between programs to solve problems that could not be solved alone [11]. Agents are being used for many reasons. Some of these reasons [12] are:

- agents can provide assistance to users in offering advice, suggestions, training, etc;
- agents can make users more productive/effective if implemented to address the needs of the users by allowing users to focus on critical tasks;
- there are lots of cpu cycles that go unused that could be working for the user while they are busy on other tasks;
- users can off-load repetitve and/or mundane tasks to their agent;
- agents are provided the knowledge of things that users should not need to know, eg. location of files or knowledge about the network;
- agents provide a good metaphor to help deal with delegation and communication in the interface.

In this paper, a generic multiagent architecture for the development of an intelligent system in power system protection is introduced. The purpose of a multiagent system is to provide each agent in the system with information and knowledge on protecting a power system. These agents are then used to assist protection engineers in the design of appropriate protection schemes for a power system or a section of it.

2.0 BRIEF REVIEW ON AGENTS SYSTEMS

There have been a big increase in the number of research going on in the development of intelligent agents in many diversified areas to assist human experts or users in their work. Such areas include monitoring, planning, designing and interfacing with other systems. This can be seen from the large number of agent based systems developed lately in multidisciplinary areas such as concurrent engineering design, industrial control, and in medical field, diagnosis, manufacturing, networking [13-20].

An example of a generic architecture is ARCHON, which has been applied to medical as well as in a number of real world DAI systems in different industrial domains. Examples of such systems are in electricity distribution and supply, electricity transmission and distribution, control of a cement kiln complex, control of a robotics application [5].

CIDIM, is another system based on ARCHON [5]. It is developed as a cooperating intelligent system for distribution management systems which is aimed to aid control engineers to ensure the continuous supply of electricity. The main jobs carried out include planning and carrying out maintenance, work safety and in coordination with field engineers, identifying faults on the network and taking necessary actions to restore supply.

Agents are also widely used in applications which provide assistance to reduce work and information overload. These interface agents provide personalised assistance with meeting scheduling, email handling, electronic news filtering and selection of entertaintment [21, 22].

In this paper, the architecture of the multiagent system consists of a number of distributed expert systems, which could be distributed

geographically or logically. These expert systems, also known as intelligent agents, are organised hierarchically and they communicate by message passing.

3.0 THE APPLICATION SYSTEM

A more complete, robust and efficient protection system should be able to design protection schemes for all components of a power system, eg. busbar, generator, motor, transformer and coordinate the schemes for any or all parts of a power system as well. In addition to that, the intelligent system must be able to provide most of the functionality required by the engineers to design adequate protection scheme for a power system or parts of it. It must also, among other things, be able: (i) to interact with the user intelligently, reason naturally, and advise the user the appropriate protection scheme for any part of a power system; (ii) to coordinate the protection schemes into a single coherent, integrated solution; (iii) to provide an explanation, if necessary, as to why a protection system is chosen.

The design of a power system protection involves making decisions which are most of the time, subjective and heuristic in nature. The knowledge required to accomplish the job cannot be acquired from textbooks and references only but also from experience as well. Although the design process involves tedious work and making heuristic decisions, it can also at times, be a repetitive and routine work especially when it comes to designing protection schemes for very similar power system. Therefore, the system to be developed must be able to assist the engineers in designing new as well as recalling previous protection schemes. To build a traditional expert system which uses just rules is not sufficient due to the limitations and inflexibility of an expert system. Hence, other alternatives must be sought and one of the better alternative is using the up and rising multiagent system. The approach adopted here is a multiagent system which employs multiparadigm[1] reasoning strategy.

3.1 System Architecture

The system architecture is a generic architecture which consists of a number of distributed expert systems, also known as intelligent agents. There are three types of agents in the system, namely Interface agent (IA), Coordinator agent (CA) and Design agent (DA) which are organised hierarchically. The hierarchical organisation supports the natural grouping of functionally related agents to facilitate cooperative problem solving. Message passing paradigm is used as the form of communication. To enhance problem solving, the agents employ multireasoning techniques including case based, rule based, explanation based and argumentation (reasoning under uncertainties). Different agents possess different knowledge about protecting different parts of the power system. This is to ensure that knowledge duplication and redundancy are avoided. Figure 1 shows the interaction of the agents in the system.

The Interface agent is responsible for making interactions and communications with the user or an external program. It interacts with the user and translates the user's request to the other agents in the system. The purpose of having the Interface agent is to free the Coordinator and Design agents from tedious communication process so that they could focus and carry out their respective responsibility effectively. The Coordinator agent receives the initial problem from the Interface agent and decomposes it into smaller tasks, which are then distributed to the appropriate Design agents. The Design agents are also known as problem solvers - each of them specialised in a particular area of the domain. They are situated at the lowest level of the hierarchy. Given a task, they formulate a solution and submit it to the Coordinator agent. The Coordinator agent accumulates and coordinates all the sub-solutions into a complete integrated solution to the user original problem. This solution is send to the Interface agent which then relays it back to the user.

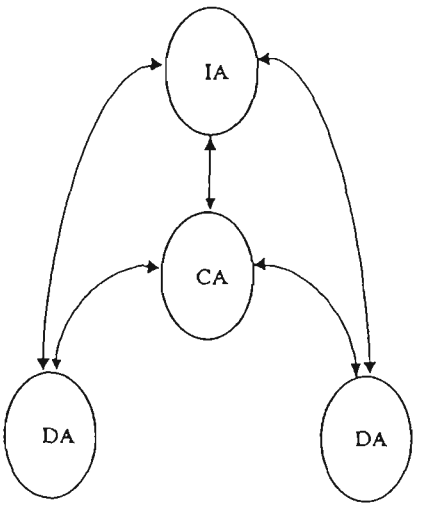


Figure 1. System Architecture

3.2 Agent Architecture

The agents have a similar architecture design which are multilayered. The layers within an agent consists of a `communication`, a `control`, a `knowledge base` and an `inference knowledge`. However, in the Coordinator and Design agents, the domain knowledge layer is split into `domain knowledge` and `case library`. These layers are organised into *conscious mind*, *logical mind* and *memory* as shown in Figure 2.

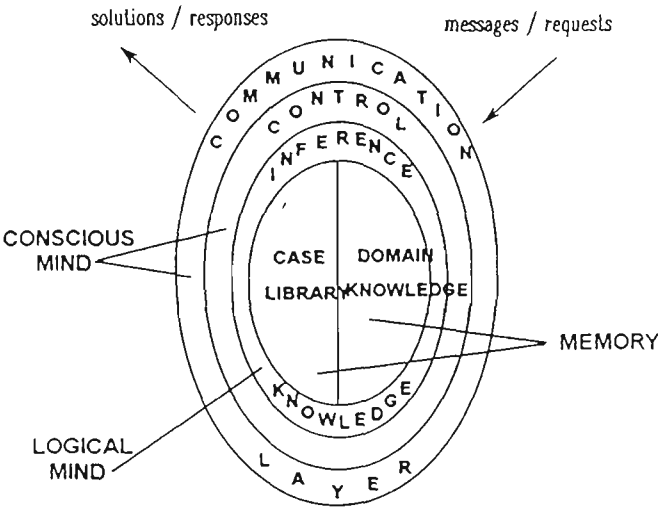


Figure 2. Agent Architecture

3.2.1 The Layers Within

Both the `communication layer` and the `control` are treated as the *conscious mind*. Together, they have the responsibility towards the whole being of the agent. To be able to function effectively and efficiently on its own, the agent needs to be able to react and respond correctly and accordingly to all different types of messages it may receive. The `communication layer` is the interface layer where the agent communicates with the outside world. It interprets the messages received while the `control` responses them appropriately with necessary actions. The `control` directs, controls and coordinate the overall functionality and operations of the system. It may schedules

[1] Multiparadigm approach is adopted because it allows the system to be more flexible, and to benefit from the utilisation of as many paradigms as it needs - each paradigm handles one aspect of the system that it is best suited for. In addition, the different paradigms or mechanisms employed can be used to complement one another and make up the limitations of the other.

the tasks and pass them to the inference knowledge via message passing paradigm.

The `inference knowledge`, also referred to as the *logical mind*, employs multireasoning paradigms to carry out its responsibility efficiently. It utilises the appropriate knowledge which is kept in memory to carry out the assigned tasks. The different reasoning mechanisms are organised into layers. Residing on top of these layers is a selector. The selector is responsible for task decomposition and calling the execution of the appropriate reasoner to solve a problem or complete a task. In other words, the selector can be said to be responsible for the automatic switching of the reasoning or inference mechanisms, whenever necessary, during problem solving.

The *memory* is organised and divided into two sections - the `domain knowledge` and the `case library`. The `domain knowledge` contains all the rules and facts about the domain, but it provides no information about how the knowledge should be utilised. Some examples of the facts in the `domain knowledge` are the current transformers (cts) requirements for a certain protection scheme eg. *a high impedance scheme requires dedicated current transformers with equal ratio*, detailed information on available relays and so on.

The `case library` contains past experiences or cases that have been previously solved by the agent. Not all cases are stored by the system - only new cases which are not similar to the existing cases in the `case library` are kept. Furthermore, the `case library` is divided into two parts - the primary and the secondary section. The primary section of the *memory* stores the cases which have either been proven by actual successful implementation or verified by a human expert. Whereas, the secondary section keeps the newly constructed cases and those cases which have been adapted and modified but have not been proven or verified as yet. Retrieval is mainly carried out from the primary memory only. In situations where there is no similar cases found, retrieval is then performed on the secondary memory. This separation and organisation of the case memory provides a more efficient retrieval and searching of similar cases.

4.0 SYSTEM OPERATION

A prototype system has been built which involves the bus and line components of a power system. The system contains some verified coordinated protection schemes in the `case library` of the Coordinator agent and some verified bus and line protection schemes in the Bus and Line agent's memory respectively. It begins with the user specification of the bus and/or the line to be protected, eg. whether backup protections are required, the cts characteristics and so on. Provided with this information, the system goes through the communication process among the agents based on message passing paradigm. The result is a coordinated protection scheme which could be a verified design or otherwise, depending on how the solution is derived. If the solution is adapted from a similar retrieved design or has been built from scratch, then the solution would not have been verified yet and cannot be treated as reliable. Verification of a case would require an expert's approval, which is crucial because the nature of the application system. Reliability is a very important factor in power system protection because of the risk involved and the undesirable consequences when a fault occurs if the system is not protected appropriately.

The control flow of the system can be illustrated in a flow diagram as shown in Figure 3. The system interacts with the user via the Interface agent. The user problem is passed on to the Coordinator agent, which then decomposes the problem to smaller and more manageable tasks. For example, the problem could be decomposed into the bus and line components. These tasks are then distributed to the Bus and Line agents respectively which would be designed individually before they are coordinated.

Each Design (ie. Bus and Line) agent uses its appropriate reasoning and inference on the knowledge base to derive the design solution (case) and conveys it back to the Coordinator agent. The Coordinator agent, upon receiving all sub-solutions from the Design agents, would coordinate them into one single coherent solution to the original problem. This final solution is then passed to the Interface agent which then communicates it to the user.

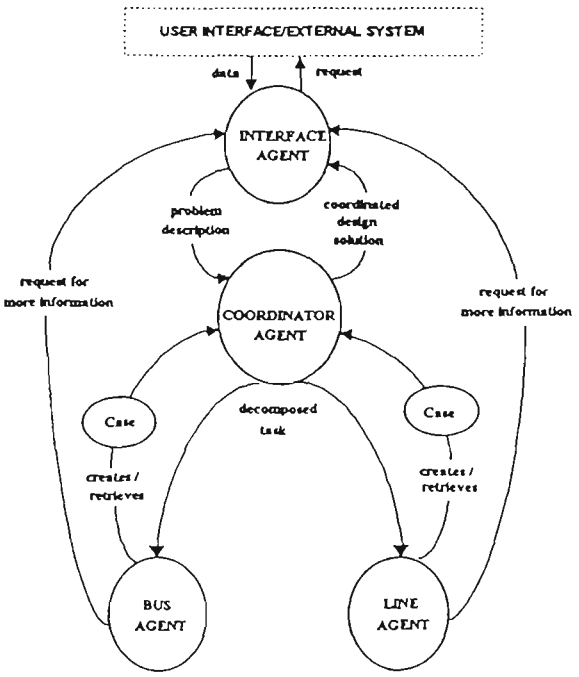


Fig 3. Control flow of the system

4.1 An Example of a Program Run

In this illustration, say, a coordinated design protection for a busbar and a line is required where the line is situated upstream of the busbar. The busbar, in return, is required to provide backup protection to an existing system located downstream of it.

The program starts with an initial interaction with the user, requesting for information about the power system for which protection is required. The IA then communicates with the CA regarding the new design. The CA then decomposes the task and sends requests to the appropriate Bus and Line agents (BA/LA) to design protection schemes for the busbar and line components respectively. Upon receipt of the request, the BA/LA will send a message to the IA requesting more information. Typical questions asked include availability of dedicated cts, possibility of cts saturating and total clearance time required.

The IA then makes contact with the original source to extract the required information which is passed to the BA/LA. Upon receiving a message from IA, the BA/LA then commence the design process. Each agent first attempts to retrieve similar designs from its memory and if successful, it will adapt the retrieved cases to the requirements of the current system. Otherwise, the agent will construct a protection scheme from first principles. When the BA/LA completes its assigned task, it then sends a reply back to the CA.

When all individual designs have been received by the CA, it attempts to retrieve similar coordinated cases from its memory, failing, it will coordinate the designs into an integrated solution. The solution is passed on to the IA, which then communicates back to the user.

5.0 SYSTEM EVALUATION

Today there is a growing demand for decision support systems which are more intelligent and can interoperate with other systems and applications. However, as the systems grow in complexity and size, they become more difficult to maintain and almost impossible to change. The multiagent paradigm presented in this paper represents a

novel approach to power system protection design. Multiagent approach promotes the development and utilisation of smaller and more manageable system components. In addition, agent technology provides a framework in which new and existing (heterogeneous) components can cooperate to achieve a common objective.

Instead of building a single monolithic expert system, a multiagent architecture consisting of several distributed expert systems which can communicate and share knowledge is developed. The advantages of this methodology are twofold: (i) smaller components are simpler and more reliable because of reduction in complexity; (ii) system decomposition aids the problem of conceptualisation and increases the system modularity, thus making the system more manageable and easier to understand.

The system knowledge is distributed among the different agents. Each agent specialises in a particular subset of the domain area. This arrangement enables agent knowledge to be updated without affecting other agents or components. Moreoever, additional agents can also be introduced relatively easily into the system. However, this is not true with traditional expert systems where changes to some rules will effect the system knowledge base. Furthermore, introducing additional components or subsystems may result in revamping the entire system.

Another advantage of the agent based approach is the ability of the agents to employ multi reasoning strategies ranging from rule based reasoning through argumentation and case based reasoning. The agent inference mechanism must be sufficiently advanced to reflect the expert's way of thinking. The ability to use multiple reasoning strategies, selecting the most appropriate strategy for the task while switching between the strategies when necessary, appears central to the success of expert reasoning. Majority of experts, including protection design experts, would normally rely on their experience when solving problems. If they have not confronted a similar problem before, they would apply their domain knowledge and arrive at a solution using first principles. This strategy has been implemented in the prototype system.

A distributed system which consists of a community of communicating and cooperating agents is far more flexible, versatile and modular compared to the older generation of expert systems. The new generation systems can be tested and expanded incrementally with minimal changes to the existing system. In addition, the knowledge which is contained in the system can be fully or partially reused when adding new system components.

Finally, the implementation of different reasoning paradigms and switching between them, knowledge sharing, incremental growth and changes to the knowledge base can be achieved more easily and naturally when using knowledge base systems. It would be difficult, if not impossible, to build in similar versatility and adaptability into the system components using traditional tools and approaches.

6.0 CONCLUSION

The multiagent paradigm represents a novel approach to the design and implementation of multiagent system to power system protection. The prototype has successfully demonstrated the use of intelligent agents in the protection area of power system. The layered agent architecture offers a more flexible and versatile approach in modelling intelligent systems. In addition, the incorporation of multireasoning paradigm enables the agents to simulate the expert's way of thinking. The architecture presented here is a generic architecture that could be applied to a wide range of application domains.

REFERENCES

[1] International Conference on Advances in Power System Control, Operation and Management, 1991, Hong Kong.

[2] Second International Conference on Modelling and Simulation, Melbourne, 1993.

[3] Australasian Universities Power Engineering Conference, Adelaide, Australia, September 1994.

[4] Bussmann S. and Muller J., A Communication Architecture For Cooperating Agents, *Computers and Artificial Intelligence*, 1993 vol.12(1), pp.37-53.

[5] Cockburn D. and Jennings N.R., ARCHON: A Distributed Artificial Intelligence System for Industrial Applications, *Technical Report 1994*, University of London.

[6] Jennings N.R., Varga L.Z., Aarnts R.P., Fuchs J., Skarek P., Transforming Standalone Expert Systems Into A Community of Cooperating Agents, *Engineering Application in Artificial Intelligence*, 6(4), 1993.

[7] Liu L., Gao Z., Yang Q., Tong W., Construction of Expert System for Designing Protection of Power Transformer, *ICPST 94*, pp.1380-1384.

[8] Kalam A., Klebanowski A., Poloni D., Knowledge Based System for Transformer Protection, *Proceedings on IEE International Conference on Advances in Power System Control, Operation and Management 1991*, pp.443-448.

[9] Wong S.K. and Kalam A.,Distributed Intelligent Power System Protection Using Case Based and Object Oriented Paradigms, *ISAP 96*, USA, pp.74-78.

[10] Uma G, Prasad B.E, Kumari O.N, Distributed Intelligent Systems: issues, perspectives and approaches, *Knowledge-Based Systems* vol.6 (2) June 1993,pp.77-86.

[11] Genesereth M.R. and Ketchpel S.P., Software Agents, *Communication of the ACM*, July 1994 vol.37(7), pp.48-53.

[12] Isbister K. and Layton T.L.,Agents in Review: Examples, Dimensions, and Issues, *Technical Report 94*, Stanford University.

[13] Huang J., Jennings N.R. and Fox J., An Agent Architecture for Distributed Medical Care, *Technical Report 1994*, University of Central Lancashire, U.K.

[14] Jennings N.R., Varga L.Z., Aarnts R.P., Fuchs J., Skarek P., Transforming Standalone Expert Systems Into A Community of Cooperating Agents, *Engineering Application in Artificial Intelligence*, 6(4), 1993.

[15] Khedro T, Genesereth M.R, Teicholz P.M, Agent-based framework for integrated facility engineering, *Engineering with Computers* vol.9 1993, pp.94-107.

[16] Peligry Y.P., An Illustration of the SHADE Concept: The Unit and Dimension Agent, *Technical Report No.KSL-94-24*, Knowledge Systems Laboratory, Stanford University.

[17] Winter R.L, O'Brien J., Wakefield R., A New Schema for the Engineering Design of Autonomous Agents in Non-Manufacturing Domains, *Proceedings of IEA95AIE*, Melbourne 6-8 June'95, pp.21-26

[18] Wittig T., Jennings N.R., Mamdani E.H., ARCHON - A Framework for Intelligent Co-operation, *Technical Report 1994*, Queen Mary and Westfield College, London

[19] Buttler J., Maintaining Quality and Quantity Production Constraints in a Multi-Agent Batch Scheduling Environment, *Proceedings of IEA95AIE*, pp.679-685.

[20] Maes P., Agents that Reduce Work and Information Overload, *Communications of the ACM*, vol.37(7) July 1994, pp.31-40.

[21] Lashkari Y., Max M., Maes P., Collaborative Interface Agents, *Proceedings of Twelve National Conference on AI 1994*, pp.444-449.

S.K.Wong (Email: wong@matilda.vut.edu.au)
Department of Computer and Mathematical Sciences
Victoria University of Technology
P O Box 14428 MCMC
Melbourne 8001, Victoria.
Fax No.: +613 9688 4050
Phone: +613 9688 4197

Dr A. Kalam (Email: ak@fees.vut.edu.au)
Department of Electrical and Electronic Engineering
Victoria University of Technology
P O Box 14428 MCMC
Melbourne 8001, Victoria.
Fax No.: +613 9688 4908
Phone: +613 9688 4265

To appear in

The International Journal of Power
and Energy Systems 1997,
The International Association of Science
and Technology for Development (IASTED)
Canada.

AGENTS AND ITS APPLICATION IN POWER SYSTEM PROTECTION

*S.K Wong, *T. Alwast, ***C. Biasizzo, **A.Kalam
*Department of Computer and Mathematical Sciences
**Department of Electrical and Electronic Engineering
Victoria University of Technology,
P.O Box 14428, MCMC, Melbourne, Victoria 8001, AUSTRALIA.
***Power System Consulting Pty Ltd.,
54, Fernside Ave, Briar Hill, Victoria 3088, AUSTRALIA

ABSTRACT

This paper presents an agent based architecture in the development of an intelligent system in power system protection. The architecture designed is applicable to various application domains. Agents present not only a better alternative to the more traditional methodologies eg. rules, but also a more flexible and versatile approach in the development of intelligent systems. Agents are represented as sophisticated expert systems which can communicate with each other via message passing. The system consists of: (i) an interface agent which is responsible for communicating with the user, (ii) a coordinating agent which is responsible for task decomposition and coordination of designs and (iii) design agent, which specialises in solving tasks in its domain. These agents are organised in a hierarchical manner. Each agent is designed as a multilayered architecture which incorporates multireasoning paradigm to enable it to switch between the appropriate reasoning techniques during problem solving. A prototype system for the design of protection schemes for bus and line components of a power system has been implemented. The system illustrates how the agents, with their respective knowledge, function and communicate with one another to design coordinated protection schemes for a power system.

keywords: agents, multiagent system, power system protection, multi paradigm reasoning, layered architecture, message passing paradigm.

1.0 INTRODUCTION

There are increasingly more developed systems and research projects in the application of intelligent systems using distributed problem solving technique and agent based architecture. These methodologies seem to be the present and future direction for developing intelligent systems in many areas including medical, manufacturing, diagnosis, networking and engineering [10, 11, 12, 16, 20, 24, 25]. Distributed problem solving technique employs knowledge sources, also known as agents or expert systems, and allows knowledge bases to be represented in various forms and to reside in different environments/platforms. In some system, multiparadigm technique is incorporated to make the system more powerful, flexible and robust.

The growth of distributed intelligent systems in the electrical and power areas has been slow. As far as literature survey reveals, there is no agent based system

developed in the area of power system protection as yet. Most of the developed systems are mainly expert systems and a small number of knowledge based systems. They are known as production rules systems that commonly employ forward or backward chaining process as the inference.

Expert system paradigm has always been popular in the power system field. The areas of interest cover power system management, system restoration, forecasting, protection, communication, distribution, transmission, scheduling, monitoring, security, control and many more. However, the number of research projects in intelligent systems in power system protection is relatively small. These include application and simulation systems developed for setting calculations for distance protection in zones 2 and 3, load forecasting, distance relaying, alarm processing, fault analysis and transformer protection. [2, 9, 13, 18, 19]. Even though rules may seems sufficient in some systems, they frequently suffer obvious setbacks especially when the systems begin to expand. Such systems are not only very brittle but the application domain has to be well defined. Modifications to the rules often cause some undesirable side effects to the knowledge base. At the same time, major difficulties arise when dealing with problems which are ill defined and, at the same time, are highly diversified. The difficulties are due to [3]:

- Building and maintaining the consistency of a complete and large rule based system;
- Integrating the various knowledge representations and reasoning strategies used by different systems;
- Restricting the problem domain of an expert to a specific area.

Hence, to build robust and flexible systems which are maintainable and expandable, production rules alone are insufficient. Other methodologies have been introduced and used to either replace or integrate with rules. Some of these reasonings include memory based, case based, explanation based, function based, goal based, experiential reasoning, reasoning under uncertainties, causal reasoning, and so on. In fact, many application systems nowadays are hybrid systems that employ multiparadigm reasoning techniques. Different reasoning techniques are used to complement one another and make up the limitations of the other methodologies.

2.0 SYSTEMS REVIEW

Most of the older generation systems employed one relatively simple inference engine. This inference, usually represented as production rules, is applied to the knowledge base which is represented in a particular format. As reported by Wielinga et. al [24], knowledge base of such systems, eg. MYCIN, hides various important properties of the reasoning process and of the structure of the knowledge in the application domain. Certain rules, or parts of the rules, remain implicit in such knowledge based systems. This implicitness impairs the acquisition and refinement of knowledge, reuse of the system, its explanatory power and the assessment of its relation with other systems.

The application systems developed in the area of power system protection have been mainly expert systems that make use of production rules. Among these applications, majority of them are implemented using an expert system shell which makes prototyping or building of a system easier. However, at the same time, there are a lot of constraints and restrictions placed on the system to be built.

SEPT [6] is an expert system which performs diagnosis task based on the comparison of the observed behaviour and the expected behaviour modelled in the knowledge base. The two main functions of the system are protection system monitoring and network incident identification. SEPT is developed using an object oriented language and its knowledge base integrates about 600 formal expert rules. But maintaining the knowledge base with numerous changes to the rules, eg. changing of equipment and their settings would be a difficult task.

Kawahara [15] presents an expert system which uses production rules to set the directional overcurrent and distance relays with regards to loop systems. Knowledge of protective engineers on the relay settings are translated into production rules. The system is built on OPS3, which is a supporting tool for building expert systems. Again, in addition to a restricted well-defined application domain, maintaining the knowledge base would also be a difficult task.

CAPE [4] is a productivity tool developed on InterBase, a commercial relational database management system. It provides comprehensive computing and record-keeping environment for activities of a system protection group. There are various modules in the system which are defined to perform certain functions. CAPE database is implemented as a fully relational and distributed database with additional relations to represent the database structure. The data and knowledge are organised and kept in tables format. However, there could be problems arising from updating the large number of tables, whilst maintaining the consistency and integrity of the database. Furthermore, the process could be slowed down by queries which needs access to several tables at the same time.

Other applications that focus on different aspects of power system protection include applications that aid protection engineers in the design of protection systems. The number of such applications are very small. Furthermore, the importance and potential contributions of such systems have somehow been overlooked. This type of application systems which aim to help the protection engineers in the design and analysis of power system protections could be very useful and practical. The functionalities of such systems may include verification of a protection system, calculating the settings of the relays and a relay database of various manufacturers. Among the few applications that have been developed in this specific area are [14, 18, 26].

Both systems developed by Liu *et al.* [18] and Kalam *et al.* [14] looked into transformer protections. The former system is made up of a knowledge base, an inference engine, a user interface and several subroutine programs for setting current transformers (cts), power transformers (pts) and relays. It is an expert system built on OPS83 using frame structures that assists the users to determine the protection system for power transformers only. While the latter system is developed using Personal Consultant Plus which plays an advisory role in suggesting the necessary protection scheme for a given power transformer setup.

Recent attempts to develop larger and more complex knowledge based systems have revealed the shortcomings and problems of centralised, single expert system architectures [1]. There are very few distributed applications in power system [2] and the growth of such applications in power system is rather slow. Moreover, these systems have been only to serve as an integration unit for a number of geographically distributed systems or different components within a system. The functions of these systems basically involve data exchange and monitoring; the reasoning technique employed is mainly inductive reasoning.

3.0 PROTECTION FOR POWER SYSTEM

A power system represents a very large capital investment. To maximise the return, the system is loaded as much as possible and is usually in full operation continuously [7, 21]. The system is always exposed to the risk of a fault occuring which could be due to storms, lightnings, falling of external objects, damage to insulators and so on. Such incidents would not only cause mechanical damages but also an electrical fault. A fault usually produces repercussions thoroughout the network. Therefore, the risk of a fault occurring, however small it may be, is multiplied by the number of items which are closely associated in the extensive system. As such, it is imperative to provide some sort of protection to the power system.

Protection for a power system is basically viewed as an integrated and coordinated sum of protection for all parts of a power system which may include busbars, lines, transformers, motors, generators, reactance and other parts of the power system that need to be protected. The main aim of a protection system is to minimise the damages and risks if and when a fault occurs. The most serious consequences of an uncleared fault is fire which may not only destroy the equipment of its origin but may also spread to other parts of the system, causing total failure and endangering lives. The applicability and suitability of a protection scheme depend on factors such as location and importance of the power system, components or parts of the power system to be protected, economic constraints, availability of resources, budget allocation, utility policies and cost of implementation.

A good protection scheme is said to be reliable and efficient if it operates within the specified setting time without fail. A protection system consists of a set of protective gears which may include circuit breakers, fuses and relays installed at various parts of a power system. The possible implications of an incorrect protection operation or even delayed fault clearance can be disastrous. Such events will progressively multiply and their undesirable consequences and effects will continuously increase.

It is important that the protection scheme is designed effectively which is the responsibility of a protection engineer. The design of protection scheme for power system is not a simple task. It represents tedious work that requires a great deal of expertise and experience on the part of the engineer. In addition, the decisions regarding protection are at most times subjective and follow 'rules of thumb'. The protection schemes for power system elements must be designed so that the system meets the reliability requirements, speed and selectivity as set by the system operating constraints [26].

In this paper, a multiagent system is introduced to assist the protection engineers in their work. The system consists of several distributed and cooperating object based expert systems also known as agents. In addition to distributed knowledge, the system also employs multiparadigm reasoning techniques for added flexibility and efficiency. The system could also be used as a training or learning tool for new graduates and inexperienced engineers in the protection area.

4.0 SYSTEM ARCHITECTURE

The system architecture, illustrated in Figure 1, consists of a group of loosely coupled and decentralised problem solving agents, ie. Interface agent (IA),

Coordinator agent (CA) and Design agent. In the prototype system, the Design agents created are the Bus and Line agents (BA and LA).

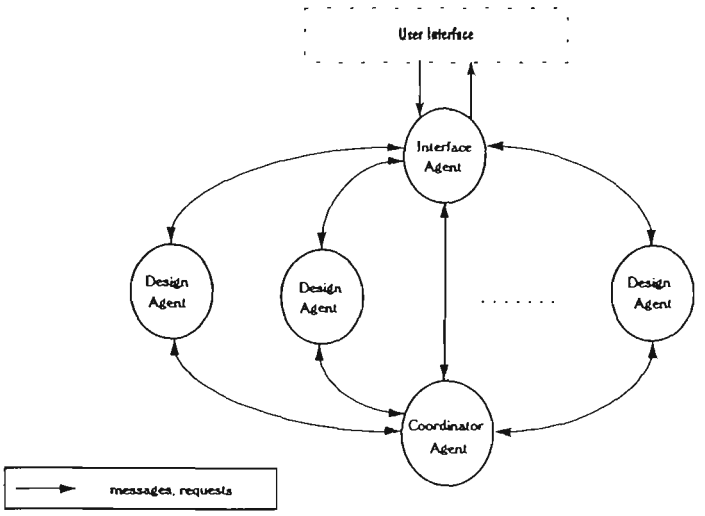


Figure 1. System Architecture

The arrows in Figure 1 show the direction of the communication flow between the agents which is accomplished by message passing using Knowledge Query and Manipulation Language (KQML)[1] expressions. The agents communicate with each other with the exception of the Design agents.

The Interface agent is built to provide an intelligent interface with the user. All communications between other agents and the user is achieved via the Interface agent. For example, if the Coordinator agent needs to communicate the final design to the user or the Design agent requires more information about a problem, the Interface agent would interact with the user on their behalf.

The Coordinator agent is responsible for decomposing an original problem, distributing the decomposed tasks to the appropriate Design agents and eventually coordinating the various design solutions into a single coherent solution. The Design agents such as Bus and Line agents, employ multiparadigm reasoning techniques to construct the solution, which would be an applicable protection scheme for the bus and line components of a power system respectively.

4.1 Agent Design

An agent is a complex artificial intelligent system which possesses substantial knowledge and reasoning components. Each agent differs in its knowledge and expertise and is constructed using a number of distinct and loosely coupled layers. The agent architecture is a multilayered structure containing sublayers within layers. The top layers, shown in Figure 2, include Control, Inference Knowledge and Knowledge Base.

The Control layer, akin to the human conscious mind, is responsible for task decomposition, scheduling the agent activities and communicating with other agents by sending/receiving messages to/from other agents. The Inference Knowledge, an analogy to the human logical mind, is organised into several layers within the Inference Knowledge layer: performative layer, function and procedural layer, and primitive layer. The performative layer defines the agent capabilities in terms of tasks

[1]KQML expressions are used to form the agent messages [8].

definitions. The various reasonings, ie. heuristics, analogical reasoning and case based reasonings [17] are implemented in both the function and procedural layer while the primitive layer contains the search algorithms.

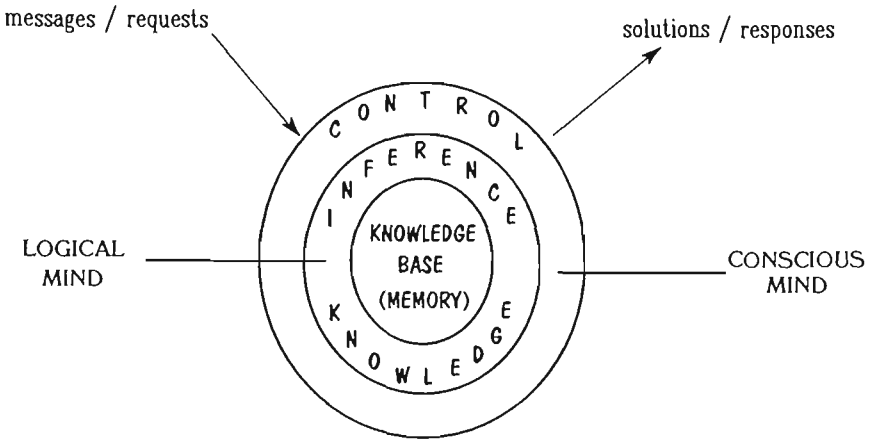


Figure 2. Agent Architecture

The Knowledge Base layer is organised into domain knowledge and case library. The case library contains previous experiences and is divided into two sections, primary and secondary. The primary memory stores the more popular and frequently used designs. Whereas designs kept in the secondary memory can be easily loaded into the primary memory when required. The organisation of the agent Inference Knowledge and Knowledge Base is illustrated in Figure 3.

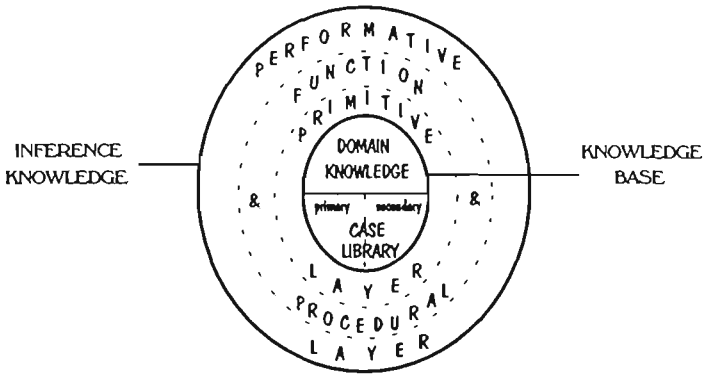


Figure 3. Agent Inference Knowledge and Knowledge Base

4.2 Agent Construction

The agent is built and equipped with the knowledge and expertise required to fulfil its responsibilty and perform its various functions, including task decomposition. The agent is constructed in Lucid Common Lisp 4.1 which is loaded with two additional libraries of Common Lisp subroutines: Epilog and Application Program Interface (API). Epilog [5] is a knowledge representation and inference system based on Prolog. The language supported by Epilog is called Simplified Interchange Format (SIF), which is a subset of Knowledge Interchange Format (KIF)[2]. Epilog includes subroutines capable of transforming KIF sentences into SIF. The API [22] provides the interface between local and external agents. API also offers services such as identification of local agents, communication with API via TCP, email and

[2]KIF are terms or sentences which form the agent communication language.

lisp, definition of performatives for local agents and maintaining connections with external agents.

Agent behaviour or functions are specified using performatives. In KQML, performative is a term which describes an operation or action. The expressive power of SIF is used to define the performatives which takes the following form :

(defperformative <performative> ((receiver (eql '<agent-name>))
<arg1> <arg2> <arg3> ...)
(.....
body of performative
.....)

For example, the retrieve performative for the Bus and Line agents is defined as :

(defperformative retrieve *((receiver (eql* 'bus*))*
coordinatorcasekb buscasekb linecasekb*)*
(.....
body of performative
.....)

The arguments to the performative are the case libraries in the agent knowledge bases. These theories are coordinatorcasekb, buscasekb and linecasekb.

Table 1 show the performatives for Interface agent, together with a brief explanation of each performative. Table 2 lists the theories residing in the agents.

Table 1. Interface Agent performatives.

Performative	**Definition**
run	starts communication with the user to design a protection scheme for a power system or a part of it.
initialise_case	obtains initial information from the user with regards to the design of the protection scheme.
start_design	requests Coordinator agent to work on the user problem.
get_details	obtains component details from user.
case_details	passes component's details to the Design agent which requested it.
display_design_case	shows the final design solution to user.
get_confirmation	gets approval from user as to whether design solution is acceptable.

Table 2. Agent theories

Agent	Theory
COORDINATOR	coordinatorcaseKB (stored cases) & relayDK (stored KB)
BUS	buscaseKB (stored cases) & busDK (stored KB)
LINE	linecaseKB (stored cases) & lineDK (stored KB)

The basic unit of communication among agents is the transfer of a message from one agent to another; its purpose is to provide the receiver of the message with some information or to have the receiver take certain actions. The agent communication language includes two main components :

- A representation language (eg. KIF or SIF) for the contents of messages;
- Communication language (eg. KQML) which consists of a set of communication primitives called performatives, aims to support cooperation among agents in distributed applications. These perfomatives enable agents to exchange and request knowledge, and to cooperate in problem solving.

The agents communicate by sending KQML packages which has information about the sender, receiver, package contents, communication mode, etc. The general form of a package is :

(package *:content* *<performative or SIF description of an action>*
:sender *<agent-name>*
:receiver *<agent-name>*
:reply-with *<identifier>*
:in-reply-to *<identifier>*
:commode *<type of communication>)*

The *:reply-with* field indicates whether the sender expects a reply to the package. And if the *:in-reply-to* field is not nil, then the current package is a reply to a previous request. It is the responsibility of the receiver to send a reply if one is expected.

The package below illustrates the message sent to the Coordinator agent by the Interface agent when it receives a request from the user to design a protection scheme:

(package *:content* '(start-design
coordinatorcasekb buscasekb linecasekb)
:sender 'interface-agent
:receiver 'coordinator *)*

5.0 SYSTEM OPERATION

A prototype system has been built which involves the bus and line components of a power system. The system contains some verified coordinated protection schemes in the case memory of the Coordinator agent and some verified bus and line protection schemes in the Bus and Line agents memory respectively. It begins with the user specification of the bus and/or the line to be protected, eg. whether backup protections

are required, the cts characteristics and so on. Provided with this information, the system goes through the communication process among the agents based on message passing paradigm. The result is a coordinated protection scheme which could be a verified design or otherwise depending on how the solution is derived. If the solution is adapted from a similar retrieved design or has been built from scratch, then the solution would not have been verified yet. Verification of a case would require an expert's approval, which is important because the nature of the application system. Reliability is a very important factor in power system protection because of the risk involved and the undesirable consequences when a fault occurs if the system is not protected appropriately.

The control flow of the system can be illustrated in a flow diagram as shown in Figure 4. The system interacts with the user via the Interface agent. The user problem is passed on to the Coordinator agent, which then decomposes the problem to smaller and more manageable tasks. For example, the problem could be decomposed into the bus and line components which would be designed individually before they are coordinated. These tasks are then distributed to the Bus and Line agents respectively.

Each Design (ie. Bus and Line) agent uses its appropriate reasoning and inference on the knowledge base to derive the design solution (case) and conveys it back to the Coordinator agent. The Coordinator agent, upon receiving all (sub)solutions from the Design agents, would coordinate them into one single coherent solution to the original problem. This final solution is then passed to the Interface agent which then communicates it to the user.

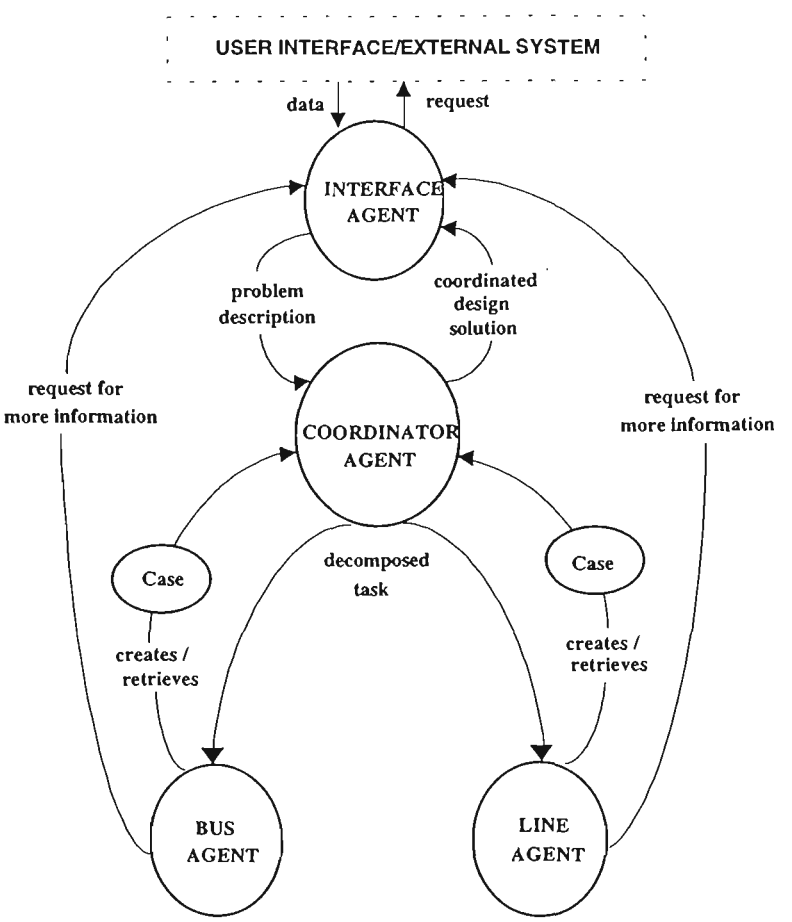


Figure 4. Control flow of the system

6.0 SYSTEM EVALUATION

Today there is a growing demand for decision support systems which are more intelligent and can interoperate with other systems and applications. However, as the systems grow in complexity and size, they become more difficult to maintain and almost impossible to change. The multiagent paradigm presented in this paper represents a novel approach to power system protection design. Multiagent approach promotes the development and utilisation of smaller and more manageable system components. In addition, agent technology provides a framework in which new and existing (heterogeneous) components can cooperate to achieve a common objective.

Instead of building a single monolithic expert system, multiagent architecture consists of several distributed expert systems which can communicate and share knowledge. The advantages of this methodology are twofold. Firstly, smaller components are simpler and more reliable because of reduction in complexity. Secondly, system decomposition aids the problem of conceptualisation and increases the system modularity, thus making the system more manageable and easier to understand.

The system knowledge is distributed among the different agents. Each agent specialises in a particular subset of the domain area. This arrangement enables agent knowledge to be updated without effecting other agents or components. Moreoever, additional agents can also be introduced relatively easily into the system. However, this is not true with traditional expert systems where changes to some rules will effect the system knowledge base. Furthermore, introducing additional components or subsystems may result in revamping the entire system.

Another advantage of the agent based approach is the ability of the agents to employ multi reasoning strategies ranging from rule based reasoning through argumentation and case based reasoning. The agent inference mechanism must be sufficiently advanced to reflect the expert's way of thinking. The ability to use multiple reasoning strategies, selecting the most appropriate strategy for the task while switching between the strategies when necessary, appears central to the success of expert reasoning. Majority of experts, including protection design experts, would normally rely on their experience when solving problems. If they have not confronted a similar problem before, they would apply their domain knowledge and arrive at a solution using first principles. This strategy has been implemented in the prototype system.

A distributed system which consists of a community of communicating and cooperating agents is far more flexible, versatile and modular compared to the older generation of expert systems. The new generation systems can be tested and expanded incrementally with minimal changes to the existing system. In addition, the knowledge which is contained in the system can be fully or partially reused when adding new system components.

Finally, the implementation of different reasoning paradigms and switching between them, knowledge sharing, incremental growth and changes to the knowledge base can be achieved more easily and naturally when using knowledge base systems. It would be difficult, if not impossible, to build in similar versatility and adaptability into the system components using traditional tools and approaches.

7.0 CONCLUSION

The multiagent paradigm represents a novel approach to the design and implementation of multiagent system to power system protection. The prototype has successfully demonstrated the use of intelligent agents in the protection area of power system. The layered agent architecture offers a more flexible and versatile approach in modelling intelligent systems. In addition, the incorporation of multireasoning paradigm enables the agents to simulate the expert's way of thinking. The architecture presented here is a generic architecture that could be applied to a wide range of application domains.

A number of enhancements have been proposed for further investigation. That is, development of additional Design agents other components of a power system such as transformer, generator and motor. The agent's domain knowledge will be expanded to include more facts; the case library will also be enhanced to automatically categorise designs which have been adapted but not formally verified.

REFERENCES

[1] Adler M., Durfee E., Huhns M., Punch W., Simoudis E.,
AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents, AAAI 1992, pp.39-40.

[2] APSCOM 91, *Proceedings on 1991 International Conference on Advances in Power System Control, Operation and Management*, IEE Hong Kong, 1991.

[3] Chi T., Kiang Y., Turban E., Distributed Intelligent Agents in Decentralised Organisations, 1993.

[4] Enns M.K., McGuire P.F., Ramaswami R., Arbor A.,
CAPE: The Computer-Aided Protection Engineering System, *Proceedings of the American Power Conference*, 13-15 Mac 92, vol.2, pp.1084-1089.

[5] *Epilog 1.0 for Lisp*,
Draft Report, Epistemics 1994.

[6] Fauquembergue P., Maizener A., Parant J.M., Perrot L., Bertrand H.,
Monitoring of Protection System Behaviour Using An Expert System Which Analyses Substations Sequential Events Recording Field Experience At Electricite De France, *DPSP 93*, pp.42-45.

[7] GEC Alsthom Measurements Limited,
Protective Relays - Application Guide, 1987.

[8] Genesereth M.R. and Ketchpel S.P.,
Software Agents, *Communication of the ACM*, July 1994 vol.37(7), pp.48-53.

[9] Gora S., Kacejko P.,
Application of expert system for determination of protective devices' settings, *Proceedings 2nd Symposium Expert System Applications to Power Systems (ESAPS)*, Seattle, WA., USA, 1989, Electric Power Research Institute, Palo Alto, CA., pp 251-256.

[10] Huang J., Jennings N.R., Fox J.,
An Agent Architecture for Distributed Medical Care, Technical Report 1994.

[11] Huang G.Q. and Brandon J.A.,
Agents for Cooperating Expert Systems In Concurrent Engineering Design, *AI Edam* 7(3) 1993, pp.145-158.

[12] Jennings N.R., Varga L.Z., Aarnts R.P., Fuchs J., Skarek P.,
Transforming Standalone Expert Systems Into A Community of Cooperating Agents, Engineering Application in Artificial Intelligence 6(4), 1993.

[13] Kalam A., Negnevitsky M.,
Knowledge Based Approach To Clearing Overloads On The Power System Plant, *Proceedings of Second International Conference on Modelling and Simulation*, Victoria University of Technology, Melbourne, Australia, July 12-14 93, pp.355-363.

[14] Kalam A., Klebanowski A., Poloni D., Knowledge Based System for Transformer Protection, *Proceedings on IEE International Conference on Advances in Power System Control, Operation and Management 1991*, pp.443-448.

[15] Kawahara K., Sasaki H., Kubokawa J., Sugihara H., Kitagawa M.,
An Expert System for Supporting Protective Relay Setting for Transmission Lines, *DPSP 93*, pp.203-206.

[16] Khedro T., Genesereth M.R., Teicholz P.M.,
Agent-Based Framework for Integrated Facility Engineering, Engineering with Computers (9), 1993.

[17] Kolodner J.,
Case based Reasoning, Los Altos, Morgan Kaufmann, 1993.

[18] Liu L., Gao Z., Yang Q., Tong W.,
Construction of Expert System for Designing Protection of Power Transformer, *ICPST 94*, pp.1380-1384.

[19] Lee S.J., Yoon S.H., Moon M.C., Yang J.K.,
An Expert System for Protective Relay setting of transmission systems, *Proceedings PICA Conference*, Seattle, WA., USA, 1989, IEEE, NY, pp. 296-302.

[20] Peligry Y.P.,
An Illustration of the SHADE Concept: The Unit and Dimension Agent, Knowledge Systems Laboratory, Technical Report No.KSL-94-24, Stanford University.

[21] Ravindranath B. and Chander M.,
Power System Protection and Switchgear, Wiley Eastern Limited 1977.

[22] Singh N.,
A Common Lisp API and Facilitator for ABSI, v.2.0.3, Logic Group, Logic Report-93-4, Stanford University.

[23] Vickers A.J. and McDermid J.A.,
An Approach to the Design of Software for Distributed Real-Time Systems, Technical Report YCS-93-21, 1993, Department of Computer Science, University of York, Heslington, York.

[24] Wielinga B., Van de Velde W., Schreiber G., Akkermans H.,
The KADS Knowledge Modelling Approach, Technical Report 1992, University of Amsterdam, Social Science Informatics, The Netherlands.

[25] Wittig T., Jennings N.R., Mamdani E.H.,
ARCHON - A Framework for Intelligent Co-operation, Technical Report 1994.

[26] Wong S.K., Kalam A., Klebanowski A., A Decision Support System Using Case-Based Reasoning in Power System Protection, *AUPEC 94*, pp.682-687.

Published in

The Sixth International Conference On
Developments In Power System Protection
University of Nottingham, United Kingdom
25 - 27 March 1997.

AN AGENT APPROACH TO DESIGNING PROTECTION SYSTEMS

*S.K Wong **A.Kalam

Victoria University of Technology, Victoria, AUSTRALIA.

1.0 INTRODUCTION

Literature research have showed that many applications developed in power system protection over the past years have been mainly in the area of alarm processing, fault diagnosis, real-time protection and safety (1,2). However, there is a notably lack of intelligent systems in the area of power system protection which deals with the design of protection schemes for a power system. The reason for this shortcoming could probably be due to the fact that these systems are not real-time systems and so, do not attract the interest of many researchers.

Recently, a surge of new interest has emerged in using of agent technology in various problem areas such as medical, concurrent engineering, speech recognition, design, planning and cooperative information gathering (3-7). In addition, it has been applied successfully in other areas such as electricity distribution and supply, electricity transmission and distribution, electricity distribution and supply network and electricity transportation management (8).

Although the keyword 'agent' has been popularised lately among the researchers in intelligent systems, it is still hard to define exactly what is an agent. There is no single universally accepted definition of an agent. However, according to Bussmann and Muller (9), an agent can be regarded as an enclosed system which has an interface through which it can interact with its environment. In other words, an agent could be a human being, a robot, a procedure or anything that could be treated separately. Agents, often referred to as distributed artificial intelligence, are often used and applied in the development of more complex and complicated systems which are decomposed into smaller subsystems.

Agent technology has also been applied as distributed problem solving technique because it offers advantages such as speed, reliability, extensibility, ability to handle applications with a natural spatial distribution, ability to tolerate uncertain data and knowledge, modularity, conceptual clarity and simplicity of design (10,11).

One example of a recent agent system applied to the area of power management is CIDIM (Cooperating Intelligent system for DItribution Management systems) (8). CIDIM aims to help control engineers to manage electricity distribution and supply networks. Other systems include ARCHON (12) which is a layered multiagent architecture that facilitates cooperative problem solving in industrial applications and AGENTS (13). AGENTS is another agent system developed by Huang and Brandon to demonstrate the use of essential constructs and strategies for communication design. The approach is based on distributed and cooperating knowledge based expert system. Its agent has the standard structure of knowledge based systems which include an inference engine, a knowledge base and a global working memory shared by all agents.

This paper introduces an intelligent multiagent system which designs protection scheme for a power system or a part of it and provides explanation to the solution it proposed. A prototype system has been built and will be presented to give a better understanding of the successful application of agent technology in the area of power system protection.

2.0 THE APPLICATION SYSTEM

Further literature research showed that there is no existing system as yet which looks into the designing and assessing the protection scheme for a power system (1,2,14-16). Furthermore, it has been noted that there exists a serious problem of shortages of expertise in the area of power system protection and/or expected to stay in the organisation, which is likely to worsen in time to come. As one of the possible solution, an intelligent agent which records the knowledge of the protection experts, thus preserving the expertise and making it readily available any time would help to alleviate the loss of protection engineers and retrain new engineers.

Designing a protection scheme for a power system requires a great deal of expertise and experience on the part of the engineer. Design decisions are mainly subjective and follow 'rules of thumb'. The protection schemes can be applied jointly or separately for reasons of reliability, selectivity, cost, discrimination and speed. Hence, the successful system should automate the design process and fulfill all these functions.

In addition, a more complete, robust and efficient protection system should also:

- provide the necessary functions required by the protection engineers to design the protection schemes for any types of a power system efficiently;

- coordinate the protection schemes into a single, coherent and integrated solution;
- provide a reasoning mechanism which advise the user of the appropriate scheme for a power system;
- provide an explanation as to why a protection scheme is chosen.

In order to accomodate the aforementioned requirements, a traditional expert system is definitely inadequate. Therefore, other alternatives were sought and the result is the application of agent technology with distributed problem solving (DPS) technique and multireasoning paradigms. The reasons being that DPS technique helps to achieve results in the presence of complexity and uncertainty through cooperation while multiparadigm approach improves the efficiency and robustness of the system.

An agent based system can be represented by a collection of agents whose objectives are to solve problems in which they have sufficient knowledge (17). While DPS is characterised by the existence of interdependencies between the decomposed tasks leading to a need for the agents to cooperate extensively during problem solving. Multiparadigm approach incorporates a number of AI techniques and methodologies such as fuzzy logic, causal reasoning, case based reasoning, is introduced as an attempt to overcome the limitations and brittleness of traditional expert systems.

2.1 The Architecture of the System

The system architecture illustrated in Figure 1 is a generic architecture which consists of a group of loosely coupled and decentralised problem solving agents, that is, Interface agent (IA), Coordinator agent (CA) and Design agent (DA). The Design agent in the prototype system is made up of the Bus and Line agents, namely BA and LA.

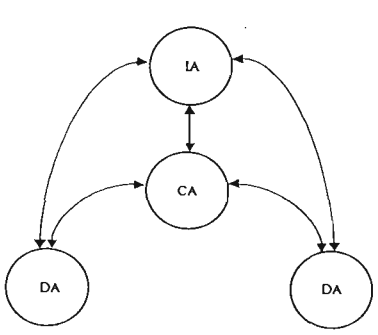


Figure 1. System Architecture

The agent in this system is a complex artificial intelligence system which is developed using the analogy of the human mind. It possesses substantial knowledge and reasoning abilities - structured on a multilayered architecture, as shown is Figure 2.

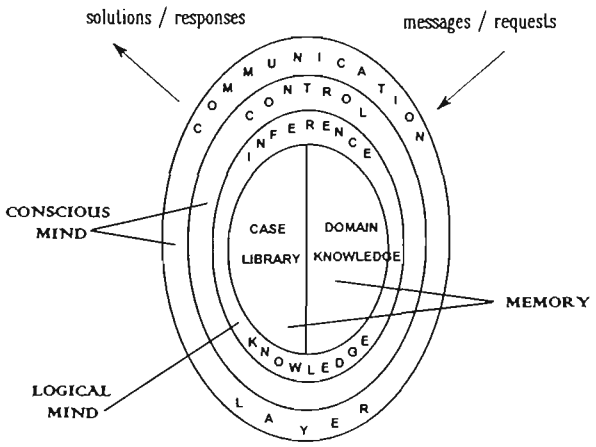


Figure 2. Agent Architecture

Both the Communication and Control layers, similar to the human's conscious mind, are responsible towards the agent's whole being. Their duties include interpretating mesages received and communicating with other agents, decomposing tasks, scheduling and managing the agent's activities. Specifically, the Control layer also directs, controls and coordinates the overall functionality and operations of the agent system.

The Inference Knowledge layer is referred to as the logical mind. It contains different reasoning mechanisms are organised into layers. A selector resides on top of these layers and is responsible for calling the execution of the appropriate reasoner to solve a problem. In other words, the selector is responsible for the automatic switching of the inference mechanisms, whenever necessary, during problem solving.

The memory is the knowledge repository area and is divided into the Case Library and Domain Knowledge. The Case Library contains expertise or experience in solving previous problems while the Domain Knowledge contains facts about the problem area. The Case Library itself is further separated into two sections - the primary and secondary. The former section stores successful past experiences (also known as cases) which have been proven in practise. The latter section stores all the newly constructed cases which have not been tested or verified yet.

The above memory organisation provides a more efficient search and retrieval of similar cases. An added advantage is reliability factor. A protection scheme for a power system has to be verified before it is applied in practise, for an inappropriate scheme could lead to disastrous results. Therefore when the solution is retrieved from the primary section, it guarantees that the solution has been verified and implemented successfully before. However, if the solution is retrieved from the secondary section, it means that the solution has not been tested before and therefore, a word of caution should be given to advise the user of the unverified solution.

3.0 THE SYSTEM OPERATION

Message passing is the basic form of communication between the agents. The purpose is to provide the receiver of the message with some information or to have the receiver take certain actions. While direct communication among the agents are permitted, communication between the Coordinator or Design agents (CA/DA) with the user is only carried out via the Interface agent. The purpose of having the Interface agent (IA) is to enable the DA and CA to perform their specific responsibilities of solving problems and coordinating the solutions respectively in a much more efficient manner. The flow of communication and system control is shown in Figure 3.

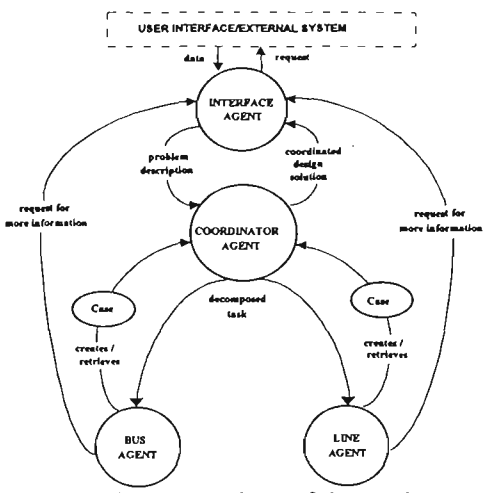


Figure 3. Flow of Control

The IA acts as a communicator between the system and the external world, which could be a user or another system. It receives messages from the external world, interprets them which could be a request or some information and passes them to the CA. A request could be defined as a problem to be solved eg. designing a protection scheme for a busbar. The CA is responsible for decomposing a problem into smaller sub-tasks and distributing them to the appropriate DAs. It also coordinates various sub-solutions into a complete integrated solution. Each DA specialises in a specific domain of the application area and applies its experience, knowledge and reasoning power to solve an assigned task.

3.1 The Prototype System

As mentioned earlier, the prototype system which has been built consists of IA, CA, Bus and Line agents (BA/LA). The system can be used to design individual or coordinated protection scheme for the bus and/or line components of a power system. The system is initially loaded with a number of verified coordinated bus and line protection schemes in the CA's Case Library and individual bus and line protection schemes in the Case Library of BA and LA respectively.

The agents are built and equipped with the knowledge and expertise required to fulfill its responsibility and perform its various functions. The agents are constructed in Lucid Common Lisp 4.1 which has two additional subroutines libraries - Epilog[1] and API[2].

3.2 A Sample Run of the Program

As an example, let's say the system is requested with a problem to design a coordinated protection scheme for a busbar which is situated downstream of a line. Further, assume both the components require backup protection.

The program begins with the problem specification by the user to the IA. This request which represents a new protection design required is communicated to the CA. The CA decomposes the task and allocates the sub-tasks to the appropriate BA and LA. Upon receiving the assigned task, the BA/LA then sends a message to the IA requesting for more information with regards to the user's requirements and the configuration of the busbar/line to be designed. Information required include availability of dedicated current transformers (cts), possibility of cts saturating and total clearance time required. The IA then interacts with the user to extract the required information and sent them to the BA/LA. On receiving the reply message, the BA/LA commences the design process. Each agent first attempts to retrieve similar designs from its memory. Any designs retrieved will be adapted, if necessary to suit the requirements of the current system. Otherwise, the agent will construct a new protection scheme from first principles. The completed task, which is a design scheme for a part of the power system is sent to the CA.

When all the involved Design agents have completed their tasks and sent their designs to the CA, the CA would then attempt to retrieve a similar coordinated design from its memory. Failing, it will coordinate the designs it received into a coherent and integrated design solution to the user's original problem. This solution is eventually communicated back to the IA for presentation to the user.

4.0 SYSTEM EVALUATION

The multiagent system presented in this paper represents a novel approach to power system protection. One of the interesting feature about the system is that it mimicks the human's natural approach to problem

---

[1] Epilog is a knowledge representation and inference system based on Prolog.

[2] API provides the interface between local and external agents, and provides identifications to the local agents, communicates with API via TCP, email and lisp, defines the agent's behaviours and functions, and maintaining connections with external systems.

solving is. applying experiences before using first principles to solve problems.

In addition, the approach promotes the development and utilisation of smaller and more manageable system components. Further, the employment of different agents to carry out domain specific tasks enables complex problems to be solved more effectively.

The system knowledge is distributed among the different agents - each specialises in a particular subset of the domain area. Such distribution allows knowledge to be updated and new additional agents to be introduced relatively easy into the system without introducing unwanted side effects. This is not true in the traditional expert systems whereby changes to the rules in the knowledge base may result in revamping the whole system.

Finally, the division of the Case Library into two sections provides not only a more efficient search and retrieval of similar cases but also, a more reliable solution. Also, knowledge sharing, incremental growth and changes to the knowledge base can be achieved more easily and naturally in a distributed knowledge base system. Conversely, it would be difficult, if not impossible to build in similar versatility and adaptability into the system components using the traditinal tools and approaches.

5.0 CONCLUSION

It has been proven as a fact that the traditional expert system is incapable of handling complex and diversified problem. As a result, more intelligent systems are being built and agent technology is being successfully applied to many complex application domains. The prototype has successfully demonstrated the use of intelligent agents which simulates the expert's approach and way of thinking in designing protection scheme for a power system or a part of it.The architecture presented here is a generic architecture that could be applied to a wide range of application domains as well.

REFERENCES

1. Second International Conference on Modelling and Simulation, Melbourne, 1993.

2. Australasian Universities Power Engineering Conference, Adelaide, Australia, 1994.

3. Hayes-Roth B, 1994, "An Architecture for Adaptive Intelligent Systems", Technical Report, Stanford University, California.

4. Huang J, Jennings R, Fox J, 1994, "An Agent Architecture for Distributed Medical Care", Technical Report, University of Central Lancashire, UK.

5. Oates T, Nagendra V, Lesser R, 1994, "Cooperative Information Gathering: A Distributed Problem Solving Approach", Technical Report version 2, University of Massachusetts, USA.

6. Peligry P, 1994, "An Illustration of the SHADE Concept: The Unit and Dimension Agent", Technical Report No.KSL-94-24, Knowledge Systems Laboratory, Stanford University, California.

7. Cockburn D and Jennings R, 1994, "ARCHON: A Distributed Artificial Intelligence System for Industrial Applications", Technical Report, University of London.

8. Jennings R, Varga Z, Aarnts P, Fuchs J and Skarek P, 1993, "Transforming Standalone Expert Systems into a Community of Cooperating Agents", Engineering Application in Artificial Intelligence 6(4).

9. Bussmann S and Muller J, 1993, "A Communication Architecture for Cooperating Agents", Computers and Artificial Intelligence, 12(1), 37-53.

10. Wielinga B, Van de Velde W, Schreiber G and Akkermans H, 1992, "The KADS Knowledge Modelling Approach", Technical Report, University of Amsterdam, Social Science Informatics.

11. Smith G and Davis R, 1981, "Frameworks for Cooperation in Distributed Problem Solving", IEEE Transactions on Systems, Man and Cybernetics, 11(1), 61-70.

12. Wittig T, Jennings R and Mamdani H, 1994, "ARCHON - A Framework for Intelligent Cooperation", Technical Report, Queen Mary and Westfield College, London.

13. Huang Q and Brandon A, 1993, "Agents for Cooperating Expert Systems in Concurrent Engineering Design", AI EDAM, 7(3), 145-158.

14. Intl Conference on Energy Management and Power Delivery, Proceedings of EMPD 1995, S'pore

15. Indnustrial and Engineering Applications of Artificial Intelligence and Expert Systems, Proceedings of IEA/AIE 95, Melbourne, Australia.

16. Intl Conference on Intelligent Systems Applications to Power Systems, Proceedings of ISAP 96, Florida.

17. Gaiti D and Pujolle G, 1992, "An Intelligent IN", International Journal of Network Management,183-189.