

University of Wollongong - Research Online

Thesis Collection

Title: Turning user into first level support in help desk: development of web-based user self-help knowledge management system

Author: Nelson K Y Leung

Year: 2006

Repository DOI:

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material.

Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.

Research Online is the open access repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

University of Wollongong Thesis Collections

University of Wollongong Thesis Collection

University of Wollongong

Year 2006

Turning user into first level support in
help desk: development of web-based user
self-help knowledge management system

Nelson K. Y. Leung
University of Wollongong

Leung, Nelson, K. Y., Turning user into first level support in help desk: development of web-based user self-help knowledge management system, M.Info.Sys. thesis, School of Economics and Information Systems, University of Wollongong, 2006. <http://ro.uow.edu.au/theses/489>

This paper is posted at Research Online.

<http://ro.uow.edu.au/theses/489>

NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

**Turning User into First Level Support in Help Desk:
Development of a Web-based User
Self-help Knowledge Management System**

A thesis submitted in partial fulfilment of the requirements for the award of the degree

Master of Information System (Research)

From

University of Wollongong

By

Nelson K. Y. Leung

Master of Information System, Griffith University
Bachelor of Information Technology, Queensland University of Technology

**Information Systems
School of Economics and Information Systems**

2006

Thesis Certification

CERTIFICATION

I, Nelson K. Y. Leung, declare that this thesis, submitted in partial fulfilment of the requirements for the award of the Degree of Master of Information Systems (Research) at the University of Wollongong, is wholly my own work otherwise I have given fully documented references or acknowledgement to the work of others. The document has not been submitted for qualifications at any other academic institution.

Nelson K. Y. Leung

May 2006

Acknowledgement

First of all, I wish to express my deepest gratitude to my supervisor, Dr. Sim Kim Lau for her patience and guidance during the course of this study. I would also like to acknowledge the support of Dr. Ann Porter who provided me the statistical consulting service. Thank you also to survey respondents, colleagues of the Commerce Research Centre as well as staff members of the Department of Information Systems, Ethical Committee and Research Student Centre at the University of Wollongong.

I would like to thank my parents who supported me in many ways unconditionally throughout the duration of this thesis. Last but not least, my warm appreciation goes to my loving fiancée Nicole. She patiently read and listened to my discoveries and even managed to look interested in which she surely had no interest. It would have been impossible to carry on this work if I did not have the warm support and love.

Nelson K. Y. Leung

List of Publications

Leung, Nelson K. Y., Lau, S. K. and Liang, G. 2005 "The Customization of Knowledge Management Techniques in Information Technology Help Desk", in the Proceedings of The 2nd International Conference on Qualitative Research in IT & IT in Qualitative Research (QualIT) 2005, CD-ROM, 23-25 Nov., Griffith University, Brisbane, Australia, page no. 11.

Leung, Nelson K. Y. and Lau, S. K. 2005 "The Development of a User Self-help Knowledge Management System for Help Desk: Deployment of Knowledge Management Approach and Software Agent Technology", in the Proceedings of The Australasian Conference of Information Systems (ACIS) 2005, CD-ROM, 29 Nov.-2 Dec., Manly Pacific Hotel, Sydney, Australia, page no. 10.

Leung, Nelson K. Y. and Lau, S. 2005 "Knowledge Management in IT Information Technology Help Desk: Past Present and Future", in the Proceedings of The 5th International Conference on Electronic Business (ICEB) 2005, CD-ROM, 5-9 Dec., Sheraton Hotel and Towers, Hong Kong, China, pp.538-545.

Leung, Nelson K. Y. and Lau, S. K. "To Ease the Dilemma of Information Technology Help Desk: A Re-distributed Knowledge Management Model", to appear in Lytras, M. and Naeve, A. Edited, Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to Their Full Potential, Idea Group Inc.

List of Figures

	Page No.
Figure 2.1 Three Levels Support Structure	27
Figure 2.2 Two Levels Support Structure	28
Figure 2.3 One Level Support Structure	28
Figure 2.4 Five Stages of Knowledge Management	35
Figure 3.1 Conceptual Knowledge Management Framework	46
Figure 3.2 Proposed Mechanism to Identify Simple and Routine Technical Enquiries	48
Figure 3.3 Proposed Re-distributed Knowledge Management Framework	49
Figure 3.4 Basic Architecture of the Proposed User Self-help KMS	51
Figure 5.1 Functionalities of the Prototype	65
Figure 5.2 Overview of the Prototype's Architecture	67
Figure 5.3 Admin and User Entry Page of the Prototype	68
Figure 5.4 Physical Design of the Dynamic Interface	69
Figure 5.5 Physical Design of the Admin Function Interface	70
Figure 5.6 Physical Design of the User Function Interface	71
Figure 5.7 Enquiry Types Category and its Partial Subclasses	73
Figure 5.8 Problem Symptoms Category and its Partial Subclasses	74
Figure 5.9 Relationships between Subclasses and Object Property (and its Inverse)	74
Figure 5.10 Partial Hierarchy of Properties and their Inverses	75
Figure 5.11 Semantic Relationships among Enquiry types, Symptoms and Properties	76
Figure 5.12 Sequence Diagram of InterfaceSoftwareAgent	77
Figure 5.13 Example to Demonstrate the Rule of the InterfaceSoftwareAgent (Dynamic User Interface View)	79
Figure 5.14 Example to Demonstrate the Rule of the InterfaceSoftwareAgent (Ontology View)	80

Figure 5.15	Sequence Diagram of SolutionRetrievalAgent and InterfaceSoftwareAgent	81
Figure 5.16	Sequence Diagram of SolutionStoringAgent	82
Figure 6.1	Admin Function Interface	84
Figure 6.2	First Sample Screen of Storing “Equipment Moving Guidance” Solution	85
Figure 6.3	Second Sample Screen of Storing “Equipment Moving Guidance” Solution	85
Figure 6.4	Sample Screen of Deleting Solution	86
Figure 6.5	Sample Screen of Retrieving Solution	88
Figure 6.6	Sample Screen of Displaying “Knowledge Unavailable” Message	89

List of Tables

		Page No.
Table 4.1	Help Desk User Base (Refer to Survey Question 1)	54
Table 4.2	Number of Help Desk Staff (Refer to Survey Question 2)	55
Table 4.3	Ratio of One Help Desk Staff to Number of Users	55
Table 4.4	Distribution of Part-time and Full-time Staff (Refer to Survey Question 2)	55
Table 4.5	Number of Operational Hours per Week (Refer to Survey Question 3)	55
Table 4.6	Help Desk Support Model (Refer to Survey Question 4)	55
Table 4.7	Help Desk Support Structure (Refer to Survey Question 5)	56
Table 4.8	Help Desk Tools and Equipments (Refer to Survey Question 6)	56
Table 4.9	Administrative Issues can be Resolved by User if Sufficient Information is Provided (Refer to Survey Question 7)	57
Table 4.10	Guidelines should be Provided to User if Needed (Refer to Survey Question 8)	57
Table 4.11	Hardware Problems User should Attempt to Solve before Using Help Desk if Sufficient Guidelines is Provided (Refer to Survey Question 9)	57
Table 4.12	Software Problems User should Attempt to Solve before Using Help Desk if Sufficient Guidelines is Provided (Refer to Survey Question 10)	58
Table 4.13	“Other” Problems Users should Attempt to Solve before Using Help Desk if Sufficient Guidelines is Provided (Refer to Survey Question 11)	58
Table 4.14	Basis of Information Provided for Question 13-18 (Refer to Survey Question 12)	59
Table 4.15	Average Number of Incoming Calls per Month (Refer to Survey Question 13)	59

Table 4.16	Average Number of Incoming Enquiries per Month (Refer to Survey Question 14)	59
Table 4.17	Increase / Decrease / No Change in Incoming Enquiries in the Past 12 Months (Refer to Survey Question 15)	59
Table 4.18	Reasons for an Increase in the Incoming Enquiries over the past 12 Months (Refer to Survey Question 15)	60
Table 4.19	Reasons for a Decrease in the Incoming Enquiries over the past 12 Months (Refer to Survey Question 15)	60
Table 4.20	Reasons for No Change in the Incoming Enquiries over the past 12 Months (Refer to Survey Question 15)	60
Table 4.21	Major Source of Contact (Refer to Survey Question 16)	60
Table 4.22	Incoming Enquiries Solved by First / Second / Third Level Support (Refer to Survey Question 17)	61
Table 4.23	Composition of Incoming Enquiries (Refer to Survey Question 18)	61

List of Abbreviations

API	Application Programming Interface
CGI	Common Gateway Interface
FAQ	Frequent Asked Question lists
HD	Help Desk
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IS	Information Systems
IT	Information Technology
JDBC	Java Database Connectivity
JSP	Java Server Pages
KM	Knowledge Management
KMS	Knowledge Management System
OWL	Web Ontology Language
RDF	Resource Description Framework
SDLC	System Development Life Cycle
SQL	Structure Query Language
URL	Uniform Resource Locator
WWW	World Wide Web
XML	Extensible Markup Language

Table of Contents

Thesis Certification.....	2
Acknowledgement.....	3
List of Publications	4
List of Figures.....	5
List of Tables	7
List of Abbreviations	9
Table of Contents	10
Abstract.....	13
Chapter 1 Background and Introduction.....	14
1.1 Introduction.....	14
1.2 Research Problems.....	15
1.3 Overview of Research.....	17
1.4 Research Aim.....	18
1.5 Research Objectives.....	18
1.6 Research Methods.....	19
1.7 Organization of Thesis	20
Chapter 2 Literature Review	21
2.1 Help Desk.....	21
2.1.1 Support Model	23
2.1.2 Service Level Agreement.....	25
2.1.3 Support Structure	26
2.1.4 Technology	28

2.2 Knowledge Management	32
2.3 Software Agent	37
2.4 Web-based System.....	40
2.5 Conclusion	42
Chapter 3 Application of Knowledge Management Techniques	44
3.1 Conceptual Knowledge Management Framework.....	44
3.2 Proposed Re-distributed Knowledge Management Framework	46
3.3 Conclusion	52
Chapter 4 Identification of Simple and Routine Enquiries.....	53
4.1 Research Methodology	53
4.2 Profile of Respondents.....	54
4.3 Identification of Simple and Routine Enquiries.....	56
4.4 Identification of Incoming Enquiry Patterns.....	58
4.5 Discussion	61
4.6 Conclusion	63
Chapter 5 Prototype Development.....	64
5.1 Design Issues	64
5.2 Development Platform.....	65
5.3 Physical Design of the Prototype.....	66
5.4 Ontology Design	71
5.5 Software Agent Design.....	76
5.6 Conclusions.....	82
Chapter 6 Prototype Illustrations	83
6.1 Illustrations of the Prototype.....	83
6.1.1 Admin Function Interface Illustration	83
6.1.2 User Function Interface Illustration	86
6.2 Conclusion	89
Chapter 7 Conclusion	90
7.1 Research Result.....	90

7.2 Research Contribution	91
7.3 Future Research	91
References.....	93
Appendix A – Information Technology Help Desk Survey	100
Appendix B – Physical Design of the Prototype	106
Appendix C – Seventy Sets of Enquiry Types and their Symptoms...	107
Appendix D – Program Codes.....	110

Abstract

Information technology has changed the way organizations function. This has resulted in reliance of help desks to support users to deal with a wide range of information technology related problems such as hardware, software and telecommunication. The help desk generally has to cover a wide range of information technology products and services. However, due to resources problem, in particular the lack of help desk staff, users often have to wait for a considerably long time before their enquiries and problems are answered and solved. Literature has shown that the majority of incoming enquiries are considered to be simple and routine which do not require specialized knowledge. This research investigates the feasibility of developing a web-based user self-help knowledge management system by applying techniques in knowledge management and software agent technology to improve the support process of routine and simple technical enquiries in the help desk. In this research, simple and routine technical enquiries are classified as problems that can be solved by users if sufficient information is provided. A survey is conducted to identify queries and problems that are considered to be simple and routine. The results also show that a decrease of incoming enquiries can be expected if sufficient online information, trainings, guidelines and technical documentations are provided to the users. A conceptual knowledge management framework has been developed to create, store, make available, use and evaluate knowledge. A prototype has been developed to demonstrate the capability of providing solutions to simple and routine enquiries. Software agent technology and ontology are applied in the proposed system. Software agent provides autonomous handling of queries and ontology formalises vocabulary in the system.

Chapter 1 Background and Introduction

This research aims to investigate the feasibility of using Knowledge Management (KM) techniques and software agent technology to develop a user self-help Knowledge Management System (KMS) in order to improve the support process of routine and simple technical enquires in Information Technology Help Desk (HD).

This chapter provides a brief introduction as well as an overview to the thesis. The chapter is organised as follows. Section 1 presents an introduction to the thesis. Section 2 describes the research problems. An overview of the research is discussed in Section 3. Section 4 presents the research aim. Research objectives and research methods are presented in Section 5 and 6 respectively. Section 7 gives the organization of the thesis.

1.1 Introduction

HD also known as computer call centre, contact centre, assist centre or support centre is an access point to provide IT-related advice, information or troubleshooting action to user. Its responsibilities include first line incident support, day to day communication between Information Technology (IT) department and user, business systems support and service quality report generating (Central Computer and Telecommunication Agency 1989, Marcella & Middleton 1996). Organizations have been investing heavily in IT and Information Systems (IS) development to solve business problems, to gain competitive advantage and to sustain organizational improvement. However, the complexity of the business systems has created infinite number of technical and functional problems. This complexity also means that users are not able to work at optimal productivity when they encounter technical problems related to the system. Organization may face potential loss in income, whether direct or indirect, immediate or in the future. The above situations have resulted in a shift of HD's role from a traditional non-profit-making function to an important management

asset that plays a vital role to ensure organizational-wide IS is working accurately and smoothly.

Unfortunately, HD is now overwhelmed by calls. It is quite common for a single HD to cover hundreds of thousands of software, hardware, application programs and network connections. Sometimes it is difficult even for the HD personnel to know all the names of hardware and software used by the organization. The adoption of management methodology such as business process reengineering and downsizing has made the situation worse. It is almost impossible for the HD to add an extra headcount to ensure that the support can be provided to users in a timely manner. Academic researchers and HD experts have continued to look for ways to relieve the above problem such as development of modern technologies, support models and structures, however the increasing workload of HD has not been improved. This research aims to investigate the feasibility of using KM techniques and software agent technology to develop a web-based user self-help KMS to help in improving the support process for routine and simple technical enquires for HD.

1.2 Research Problems

Generally, HD is divided into hierarchies so that incoming enquiries can be coordinated in an effective and efficient manner. Most HDs exploit either two levels or three levels support structure where first level support operator who has less experience and technical knowledge, will attempt to solve as many incoming enquiries as possible. If first level operator cannot resolve the problem, it will be escalated to second or third level support who possesses in-depth IT expertise. What make HD struggles is the continuous expansion of user base and the fact that HD has to cover more and more software, hardware, network and other IT related areas. It is not unusual for a single HD to cover hundreds of thousands of IT related products. On the other hand, downsizing and business process reengineering has led to the shrinkage of the size of HD because its overall budget has been reduced. This not only reduced a significant number of experienced HD staff, it has also led to the loss of priceless knowledge which is considered crucial for daily operation within the HD

boundary. When HD is expected to provide more service with less staff, the outcome is quite obvious: user has to wait comparatively longer before the first level operator is available to pick up the call. According to a recent research conducted by the Help Desk Institute (Broome & Streitwieser 2002), most respondents in the HD industry have reported their call volume has been increasing every year for the past ten years. Heckman and Guskey (1998) confirm that “help unavailable when needed” is one of the major reasons for service delivery failure in the HD, which in turn leads to user dissatisfaction. However, Knapp and Woch (2002) indicate that 80% of calls made require no specialized knowledge. Dawson and Lewis (2001) point out that close to 50% of calls to the HD at Deakin University are related to login name and password. Both researches indicate that a majority of technical enquiries and difficulties can be classified as simple and routine. As a result, HD staff are no longer available for high level and proactive support activity or training because their time are mainly occupied by answering these simple and routine enquiries. Although HD experts and academic researchers continue to look for ways with the purpose to relieve the above burden, some of their efforts, include development of systems, support structures and models for the HD, have not resulted in any improvement, moreover, the hard work seems in vain.

Human always uses reflective design concept as a method to develop a system. In other words, we tend to solve a problem based on past experience and conscious reflection without local adaptation. For example, the New South Wales Government tries to improve access to Sydney Airport, Port Botany and the Sydney City for people living in the west and south west of Sydney by building M5 East. However, the M5 East itself is actually creating congestion problem, more than 100,000 vehicles a day travel on the M5 East. This almost doubles the Roads and Traffic Authority’s calculation in its environment impact statement, predicting that 55,000 vehicles would be using the tunnel by 2011 (Smith 2005). This example shows that rather than alleviate congestion, the M5 East itself encourages more people to drive more often which in turn turning 7.1% of passengers away from the East Hill Rail Lines (Smith 2004). Similarly, various support models, structures and technologies are designed to ease high volume of enquiries within the HD environment, however, such actions actually create more troubles in the real world if the problem domain and user’s need are not investigated thoroughly. Hence, this research aims to find a suitable solution

to mitigate the overwhelming simple and routine incoming enquiries from contacting the overloaded HD.

1.3 Overview of Research

As mentioned in the previous section, a significant proportion of incoming technical enquiries from the users are usually general, simple and routine. The enquiries can be easily resolved by first level support operator and require no expertise or specialized knowledge. This research proposes to develop a web-based user self-help KMS to allow the users to solve their own simple and routine problems. The aim of developing such a system is to free up first level operator in the HD for more challenging tasks. This way when users are faced with simple technical difficulties, they can access the KMS and search for the most appropriate solution directly. In addition, the proposed KMS applies modern web and software agent technologies as a means to deliver the system. Simply by clicking on the related Uniform Resource Locator (URL), the proposed KMS will be delivered through Internet and the agent will facilitate interaction between user and the system whereby the most appropriate solution will be delivered.

Technical knowledge required to solve user's incoming enquiry usually exists either in the form of explicit or tacit knowledge. This research proposes the use of KM techniques that include create, store, make available, use and evaluate, to manage tacit and explicit knowledge in the HD. Tacit knowledge is personal, complex, hard to communicate and formalize because it is gained through individual insights overtime and is resided in human, mind and body (Martensson 2000, Nonaka et al. 2001). In contrast, explicit knowledge is structured, relatively simple and can be captured, recorded, documented, codified and shared using formal and systematic language (Goh 2002, Nonaka & Takeuchi 1995). To exploit the knowledge, externalization is required to convert the tacit knowledge such as skills, techniques, experiences and perceptions into explicit knowledge whereas combination can be used to combine and revise explicit knowledge from manual, guidelines and training documentation into one that is systematical. Nonaka et al. (2001) define externalization as a process of

making tacit knowledge into explicit knowledge and combination as a process of merging and editing explicit knowledge from multiple sources into a new set of more complicated and systematic explicit knowledge. In this way, tacit and explicit knowledge are converted in a form that can be stored and retrieved from the knowledge database within the proposed web-based user self-help KMS.

This research also proposes the use of software agent technology as an application to facilitate communication and be able to retrieve appropriate knowledge in accordance with user's need. Software agent is a computer program that behaves like human and is capable of autonomous actions in pursuit of specific goal (Liu et al. 1999, Nienaber & Cloete 2003). Software agents, with the ability to communicate and act autonomously, will be developed for the system. The ability to act autonomously relieves user from onerous searching duty by dedicating the software agent to look for the most suitable solution in the extensive knowledge database based on user's requirement. In addition, software agent is also in charge of facilitating user communication based on vocabularies stored in the ontology. It allows users to describe and identify their enquiries and their related symptoms based on hierarchy structure. Most importantly, ontology provides a shared understanding of a domain that contains a finite list of terms and their relationships (Antoniou & Harmelen 2004, Gruber & Olsen 1994).

1.4 Research Aim

This research aims to investigate the feasibility of developing a web-based user self-help KMS using techniques of KM and software agent technology to improve the support process for routine and simple technical enquires in the HD.

1.5 Research Objectives

The objectives of this research are as follows:

- 1) To investigate the feasibility of developing a web-based user self-help KMS to improve the support process for routine and simple technical enquires in HD.
- 2) To investigate the application of KM techniques to develop a re-distributed KM framework and user self-help KMS.
- 3) To investigate the application of software agent technology to facilitate communication and retrieval of knowledge in the proposed system.
- 4) To investigate the application of ontology to formalise the vocabularies of HD in the proposed system.

1.6 Research Methods

This research is conducted in the following stages:

- A survey, in the form of an online questionnaire, has been conducted to identify the routine and simple technical enquires in HD. The survey collects data on the formation of HDs and their incoming enquiry patterns. We have invited thirty-six universities in Australia plus subscribers of the Association for Information Systems World Net (AISWorld Net) by email to participate in the survey.
- A proposed conceptual KM framework is developed to identify how knowledge is created, stored, made available, used and evaluated in the HD environment. The proposed conceptual framework provides a way to manage knowledge, however it is not able to ease the overloaded HD from high volume of incoming enquiries. Since simple and routine enquiries made up of a significant proportion of incoming enquiries, we will customize the proposed conceptual framework so that a customized framework can identify and re-distribute simple and routine enquiries. This way, simple and routine enquiries can be re-distributed to the proposed user self-help KMS to allow users to resolve their simple problems by retrieving the most suitable solution from the proposed system.

- Finally, a prototype of a web-based user self-help KMS will be developed to demonstrate its capability to provide solution for simple and routine enquiry. The prototype applies software agent technology to facilitate user communication and enhance search capability. The software agent acts autonomously and will search for the most appropriate solution in the knowledge database based on user's enquiry. A communication agent is designed to retrieve the appropriate vocabulary from the ontology. This allows users to formalise the enquiries and related symptoms.

1.7 Organization of Thesis

The rest of thesis is organised as follows. Literature review related to this research is presented in Chapter 2. It includes discussion of literatures in HD, KM, software agent technology and web-based system. Chapter 3 discusses the application of KM techniques to create, store, make available, use and evaluate HD knowledge. This chapter also presents a proposed KM framework that can re-distribute simple and routine enquiry from HD to the proposed web-based user self-help KMS. Chapter 4 presents the survey results that identify routine and simple technical enquiries in HD. Chapter 5 discusses prototype development of the proposed system. It includes a discussion of the development issues, development platform, physical design of the prototype, ontology and software agent design. Illustrations of the prototype is presented in Chapter 6. Finally, Chapter 7 concludes the thesis and future research direction is proposed.

Chapter 2 Literature Review

This chapter provides theoretical background related to this research. Literature in relation to HD, KM, software agent and web-based system will be discussed.

The chapter is organised as follows. Section 1 discusses the HD support structure and model. Issues on service level agreement and HD technology are also discussed. Section 2 provides an overview of KM which includes the discussion of creating, storing, making available, using and evaluating knowledge. Section 3 introduces software agent technology and its characteristics. Section 4 discusses web-based system and its related issues. Conclusion follows in Section 5.

2.1 Help Desk

Organizations have been investing heavily in developing IS and IT (Kraemer et al. 2000) because these developments enable them to solve business problems, to gain competitive advantage and to sustain organizational improvement (Hammer 1997, Robson 1997). Consequently, the variety and complexity of software, hardware and network technology have increased substantially. This leads to the establishment of IT HD to provide technical support to users.

There is no sufficient evidence to show when the first HD was established, however HD pioneer Howard Kendall (2002) believes it has only been established for about twenty years. Before HD emerged, users either called whoever they knew or the so called “computer expert” in the IT department when they required technical support (McKoen 2000, Smith 1996). However, this ad-hoc support framework has some shortcomings. Firstly, IT staff might not be available for immediate assistance because they were usually occupied with other crucial projects (Prescott et al 2001). Secondly, excess amount of support duty would lead to high level of frustration within the IT department because they were not able to spend time on their own tasks or projects (McKoen 2000). Thirdly, users may often call the wrong person,

workgroup or even department for assistance (Peters 1993, Smith 1996). This would frustrate user because s/he was required to make another call or be transferred to another staff who was responsible for solving the problem. The solutions, needed to solve the problem not only were delayed the support process, may also interrupt the development and deployment of new services and systems in the IT department. Thus the idea of HD began to emerge with the purpose to minimize the above problems and to meet user's expectation.

HD functions as an access point to provide IT-related advice, information and troubleshooting action for user. It also acts as a facilitator to collect and analyse data that can transform itself to a more proactive role (Marcella & Middleton 1996). The Central Computer and Telecommunications Agency (1989) stresses that the responsibilities of HD include first line incident support in case of IT failure, day-to-day communication between IT department and user, business systems support and service quality report generating. Workman and Bommer (2004) cite the importance of HD as to provide technical assistance to users in case of computer-related hardware or software failure. In short, it is a first contact place for user relating to all IT support issues. Generally, IT related support issues include:

- 1) Software / application / hardware / data communication device/ telecommunication device usage enquiry
- 2) software / hardware / data communication device / telecommunication device installation
- 3) repair, troubleshoot and configuration
- 4) user account setup
- 5) security issue
- 6) Internet / email support
- 7) service / product purchasing
- 8) inventory management
- 9) training

2.1.1 Support Model

Decentralized HD model was very popular in 1980s. In this model, organisation often has more than one HD where various HDs were established by departments, branches and IT work groups (Grajek et al. 2002). For example, there were nine different HDs in Western Kentucky University (Kirchmeyer 2002). Within the university, user had to determine which HDs to call, depending on where the problem was, what the problem was and when the problem occurred (Kirchmeyer 2002). The decentralized model shared the belief that diverse support issues could be referred to related HDs easily so that timely response could be acquired. This concept worked well at the very beginning because computer system was simple. At that time, the structure of the computer system was straight forward and consisted of only dumb terminals, mainframes, printers and simple stand alone application programs. As IT infrastructure became more complicated, organization-wide systems with a large number of interconnected hardware and software, classification of problem domains became less distinct. In such situation, users were confused with multiple HDs and were often required to be transferred from one HD to another before obtaining a correct solution (Middleton 1999). Fortunately, HD evolution just kept going. In order to restore its reputation, organizations started to adopt centralized HD model. The idea is to merge various HDs into one and user only needs to remember one contact number for all IT related queries which makes HD the first and single point of contact (Middleton 1999). This model not only consolidates the contact point, it also helps to consolidate and standardize diverse support policies and procedures, service level agreements as well as HD support tools (Kirchmeyer 2002, Middleton 1999). Other incentives for this model include better resources allocation (Greenberg 1998), improve resolution rate and inter-division communication (Scullen 2001).

Nowadays, some global corporations, with offices located all over the world, implement another concept called distributed or virtual HD model. Though this model promotes HD of multiple physical locations, user can still contact the HD by using one contact number through the modern call routing technology (Tischler & Trachtenberg 1998). In this way, HD is able to operate twenty-four hours a day, seven days a week regardless of location. For example, Morgan Stanley, one of the largest investment banks in the world, consists of four HDs in different sites (USA, England,

Japan and Hong Kong) that enable them to provide enterprise-wide twenty-four hours HD service. Currently, HD is further categorised as internal or external. The former only supports organization-wide users whereas the latter supports external customers and is usually established by software and hardware vendors or Internet service providers (Heckman & Guskey 1998).

Apart from different support models mentioned above, it is also important to discuss the current trend on outsourcing of HD. IT outsourcing is not a new phenomenon. In 1960s, data processing service was contracted to vendor because of its size and cost of the required computer hardware (Lee et al. 2003). In 1970s, organizations started to build or buy their own systems but they still relied on outsourcing because there was not enough qualified IT manpower (Lee et al. 2003). The wave of outsourcing seemed to slow down afterwards but it soon resurfaced in 1990s (Lee et al. 2003). Senior IT managers are likely to outsource functions that are immaterial to core business such as HD (Kolawa 2004). The reasons commonly cited for this decision include: 1) in-house IT expert should focus on long term strategic infrastructure planning instead of servicing routine troubleshooting duty, 2) outsourcers can do better job than in-house HD because they are equipped with the latest skill and technology, 3) it can increase HD productivity, efficiency and effectiveness which will lead to cost reduction, and 4) IT manager can be freed from human resources issues such as difficulty in recruiting experienced HD staff, the need to maintain sufficient staff in peak hour and so on (Faulks 2004, Gurbaxani 1996, Ketler & Willems 1999, Nam et al. 1996, Oza et al. 2004). Dash (2000) reports that the worldwide outsourcing market in HD and technical support would exceed three and a half billion U.S. dollars in 2002. Senior management no longer debates whether to outsource HD. Its major concern is the degree of outsourcing - should it be full or partial, permanent or temporary, onshore or offshore, single or multiple vendors (Krishna et al. 2004, Lee et al. 2003). Other considerations in outsourcing include data security, loss of control, loss of expertise and loss of flexibility. It is important to note that not every outsource project returns in triumph. To eliminate risk and increase transparency, organisation must build a strong alliance with outsource service provider.

Another innovative model in the HD industry is e-support. This model is gaining widespread use due to its ability to provide better, faster and cheaper service (Broome

& Streittwieser 2002). Broome and Streittwieser (2002) describe all support actions that use Internet or web as the primary communication channel to be included in e-support. One of the key stimuli in promoting e-support is the emergence of web-based tools. Users make use of email or web form to contact HD, enabling them to ignore its actual service hours. In addition, users can access online resources, such as knowledge base and Frequent Asked Question lists (FAQ), to look for information that is useful to resolve their existing difficulties. Furthermore, HD analyst is able to conduct web-based training or even using remote control technology to ease user's struggle. Although the potential of e-support is far beyond that, HDs that attempt to implement e-support model must examine carefully if the current culture, resources and technology within the organizations are ready for such a deployment (Broome & Streittwieser 2002).

2.1.2 Service Level Agreement

Surveys, questionnaires, interviews and advising committees are common qualitative methods to evaluate HD service (Sundrud 2002). The above methods certainly can provide some sorts of statistical figures but the result can be meaningless unless there is a standard to determine the degree of successfulness. Service level agreement is designed to deal with this issue. Hathaway (1995) defines service level agreement as a contract like document that describes user requirement on service level and the scope of support that HD is offering. Niedzwiecki and Peterson (2002) emphasize service level agreement as a tool: 1) to clarify rights and obligations for users and HD members, 2) to manage user expectation and 3) to enhance bilateral communication. The first step for setting up service level agreement is to arrange meeting between HD staff and users to define requirements and expectations (Andress 2001). Though format may vary, the content of service level agreement should basically include provider of service, recipient of service, availability of service, service access method, scope of service, description of service, cost of service, user responsibility, service priority, response time, escalation procedure and reporting (Czegel 1999, Hathaway 1995, Middleton 1999, Niedzwiecki & Peterson 2002).

2.1.3 Support Structure

Each HD is unique depending on organization's strategic investments, support doctrine, business it supports and customer expectations. Generally HD is divided into front line (first level), second and third levels support (Czegel 1999) as illustrated in Figure 2.1. Kajiko-Mattsson (2003) further elaborates these three levels support structures into six variants. Instead of naming them as first, second and third level, she labels them as HD Process, Product Support Process and Maintenance Execution Process respectively. Basically, enquiries come into the front line (first level) from various sources. At this level, the first level operator will attempt to provide answers to simple questions. Users can choose to access HD through various channels which include telephone, web forms, email, fax or walk in. Most of the queries at this level are straightforward. For instance, close to 50% of calls to ITS HD at Deakin University are related to login name and password (Dawson & Lewis 2001). If first level operator cannot resolve the problem, it will be escalated to the second or third level. Second level analyst, who possesses more in-depth IT knowledge, will conduct a series of research and testing to solve the problem. If it involves on-site support such as hardware installation, second level engineer usually takes over the job. If second level analyst still cannot handle the problem, then the case will be passed to the third level specialist such as database administrator, website developer or vendor to solve the problem.

Figure 2.1 Three Levels Support Structure

Kajko-Mattsson (2003) reports that three levels concept currently dominates a large segment in HD support structures but some organisations choose to simplify it into two levels. Tourniaire and Farrell (1998) move up to categorize two levels structure (Figure 2.2) into frontline/backline approach, and “touch and hold” approach. Fundamentally, both approaches require frontline staff to handle as many enquiries as possible. As long as the problem is out of frontline’s ability and knowledge, it will be escalated to backline staff for additional investigation. The only difference is frontline operator in “hold and touch model” will be the only communication channel to user even though backline staff has taken over the problem. Other support approach includes one level support structure (Figure 2.3) which combines all three support levels in a single layer but this structure is rarely used by organization (Kajko-Mattsson 2003).

Figure 2.2 Two Levels Support Structure

Figure 2.3 One Level Support Structure

2.1.4 Technology

To support different users, HD should be equipped with high technology equipments to ensure efficient and effective troubleshooting. Fully loaded HD is never a by-product of sudden universal explosion, rather the transformation takes a long period of time with a lot of resources and efforts. According to Kendall (2002), HD in the mid-late 1980s only consisted of a desk, a phone and a pen. At that time, senior management executives never recognized the value of HD. On the other hand, HD was viewed as a non profit-generating function that always showed up as a cost on the ledgers (Czegel 1998). However, senior executives soon realized the existence of HD was essential to cope with highly-demanding users (McLay 2003). Marcella and Middleton (1996) again emphasizes HD has the potential to act as the nexus for the full integration of IT and customer service into the organization. This will lead HD to become an important management asset and increase its strategic recognition in an organization (Marcella & Middleton 1996). Middleton (1999) identifies how HD can contribute to numerous business processes while Kundtz (1996) extends this idea by applying HD to business process method. When senior managements realize HD can align with business objectives, they start to invest strategically on HD tools.

Additionally, modern technology has also accelerated the delivery of HD evolution. High-technology tools have been used to support, stimulate and accelerate the consolidation of multiple HDs. Childe, Maull and Bennett (2001) agree that a number

of business process reengineering programs are being driven by technology. Reformation never halts for HD and its present role is to be proactive rather than reactive (Cruess 2002). Marcella and Middleton (1996) further elaborate that HD is required to “fix the leak before storm” rather than “putting a bucket under the leak in bad weather”. In other words, we should try to fix all possible problems to prevent user from calling HD. The HD in Deakin University is an excellent role model of exploiting technology as a vehicle to transform itself into a more proactive HD (Dawson & Lewis 2001). For instance, the introduction of network monitoring technology at Deakin University allows the HD to proactively monitor its network services. In the following section, some of the significant HD tools will be described (Dawson & Lewis 2001).

2.1.4.1 Automatic Call Distribution System

Automatic call distribution system plays an important part in promoting HD consolidation because it can handle a large number of calls simultaneously on a single phone number. Automatic call distribution system is a system that helps to manage the flow of phone calls, record historical data and generate call statistic report (Underwood et al. 2003). When user calls, the automatic call distribution system that interconnects a finite number of HD operators, will distribute the call to the first available operator. If all operators are busy, the call will be placed in a queue. Most of the systems will then play a recorded message to inform user that “all lines are currently busy and the first available HD operator will answer the call as soon as possible”. At the same time, the automatic call distribution system keeps monitoring the queue, sending the first user in the queue to the next available operator and makes sure the calls are evenly distributed among the HD operators. An interactive voice response system is widely installed as a front end for the automatic call distribution system. The interactive voice response system is an automated answering system that allows user to interface with other technology such as mainframe, database and fax machine. It also allows the users to get information or to perform a specific function simply by selecting the required options from the menu via the telephone pad (Czegel 1999). Additionally, the automatic call distribution system that possesses supervisory function enables HD supervisor to monitor the workload, listen-in to calls, monitor queue status, re-route calls and re-configure automatic call distribution system

settings to fit different call patterns (Bornhoft et al. 1991). Supervisory and management reports that include total incoming, outgoing plus abandoned calls, call answered, average talk time and average hold time can be generated by the automatic call distribution system (Czegel 1999, Underwood et al. 2003). These reports allow HD to continuously enhance its performance by re-arranging manpower, purchasing or developing new technologies or changing automatic call distribution system configurations. For instance, if the statistic shows there are an enormous number of abandoned calls in the morning, then more staff should be added to the morning shift.

2.1.4.2 Help Desk Management System

The emerging of HD management system is a major step for HD automation (Middleton 1999). Czegel (1998) depicts four basic functions of HD management systems as call information logging, ticket escalating, ticket storing as well as reporting. Call logging function enables the HD operator to record user's personal detail, computer setting, and problem description in a ticket storing function or ticket repository. The HD staff always refer to that piece of record as a ticket. As soon as the user calls to request technical support, the HD operator has to open a ticket, fill in the details and then save it in the storing function. If the problem requires further escalation, the operator can forward or assign the ticket to a particular analyst or workgroup by the ticket escalating function. Analyst or workgroup who holds the ticket is responsible for updating all follow-up action, progress and resolution method into the ticket repository. When the problem is resolved, the ticket will be closed. The reporting function allows HD supervisor or manager to generate report with different parameters, such as high priority ticket, outstanding ticket, problem type and so on (Underwood et al. 2003). Reporting is a very powerful function to manage the daily operation of HD. For example, if there are too many outstanding tickets waiting to be resolved, it maybe an indication to hire more staff. In another instance, if there are a huge amount of tickets related to a software or hardware problem, then it may require a thorough check up on the system concerned.

2.1.4.3 Rule Based Expert System

Expert system has been highlighted as a feasible application in the HD industry due to the scarceness, diverseness and expensiveness of expertise (Abraham et al. 1991, Goker et al.1998, Goker & Roth-Berghofer 1999). The ever and fast expansions of IT often result in the HD staff require specific knowledge and expertise to understand and handle the enterprise-wide system. Consequently, it makes the HD staff impossible to offer immediate assistance if one of the experts with a particular knowledge is unavailable. Expert system or knowledge-based expert system is a subset of artificial intelligence which imitates human reasoning process to solve specific problems (Turban & Aronson 2001). Giarratano and Riley (1998) use the word “emulate” to describe the intention for an expert system to act and make decision like or even better than human. If an expert system is developed, the first level operator is able to provide recommendation and solution for a routine or even complex problem simply by entering its description plus symptom to the system. Then the embedded inference engine will try to find the best diagnostic method from the knowledge-based system. This way, the second and third level support staff can be freed for more important duty. Expert system ensures not only the availability of expertise but also minimize the problem solving duration and cost. However, Middleton (1999) argues that expert system and other artificial intelligence related system are not as widely used as expected. Some of the problems in developing HD expert system are high cost and time consuming in knowledge acquisition as well as knowledge base maintenance, high complexity of problem domains, not user friendly and difficulties in HD expert system development (Czegel 1998).

2.1.4.4 Remote Control

Remote control is a HD software that makes use of modern data communication technology to view, access or even take control of computer to carry out troubleshooting over the network (Rea & Cleary 2001). There are two types of remote control software: client-based and web-based. The only difference is that client based requires installing a small program called client, whereas web-based simply connects through the Internet. Compared to traditional on-site support method, remote control provides a quicker way for problem solving as long as the target computer has

Internet access and it also encourages user's involvement in fixing a problem by watching and learning the required process through the technician's demonstration. However, security is always an important issue with remote access. Auspiciously, most of the software can be configured so that the technician must gain permission from the user before viewing and controlling the target computer. Additionally, user can re-take control or even terminate the session at any time.

2.2 Knowledge Management

Human is able to dominate the world despite their physical weakness as compared to other animals. It is mainly the contribution of our intelligent ancestors who created sufficient sets of survival skills. Through hundreds of thousands years of evolution, they not only improved those vital knowledge, but also transferred them from generation to generation by various communication methods, first verbalism, then cave drawings, then alphabetic and text writing on clay tablet and papyrus, finally modern language via contemporary recording media and devices (Ives et al. 1998, Yule 1996).

Back in mid 1980s, management tools and techniques such as total quality management, downsizing and business process reengineering were developed by western companies to aid in re-gaining market share in automotive and electronic appliance industries invaded by the Japanese companies (Chase 1997). However, both input and improvement are short-term because these solution approaches are generic and easily available to all rival companies (Sharkie 2003). Once an approach is proven successful, the rival company duplicates and adopts the same practice (Sharkie 2003). The practices of downsizing, outsourcing and business process reengineering, which aim for process optimization as well as cost and time saving, have resulted in the loss of many experienced employees along with their capability and knowledge which have in turn taken away organization's priceless inspiration and creativity (Coulson-Thomas 1997). Hence, organizations have to pay high, severe and long-term price in return for transient benefit. The worst is after several years of downsizing and business process reengineering, companies in the western world are now competing

with each other on equal cost, quality and delivery performance levels (Chase 1997). This means the company has difficulties in differentiating with their competitors. What intensify the already fierce battlefield is the availability of cheap labour in Asian and other developing countries (Chase 1997). Thus, the concept of KM is emerged to sustain long term competitive advantage by preserving organizational knowledge (Turban & Aronson 2001). Knowledge is now recognized as one of the most important management assets because knowledge enables organizations to utilize and develop resources, enhance their fundamental competitive ability and develop sustainable competitive advantage (Sharkie 2003). In other words, knowledge allows an organization to do better than its rivals.

Before continuing the discussion of KM, it is essential to clarify the meaning of knowledge. Knowledge is not an uncommon word. In a study, 92% of respondents claimed that they worked in knowledge-intensive organizations (Chase 1997), however, many people still confuse the differences among data, information and knowledge. Data are raw facts, whereas information is data that has been refined, processed and organised to support decision (Rob & Coronel 2002, Whitten et al. 2001). Smith (2001) adds that most data is in the form of numeric, basic information or observations of work activities that can be quantified while information is data with relevance, purpose as well as context. Information has little value until human intervention is applied to extract its meaning or use on the job. On the other hand, knowledge appears in forms of facts, attitudes, opinions, issues, values, theories, reasons, processes, tools, relationships, risks and probabilities. Knowledge is often considered as information that contains specific properties (Coulson-Thomas 1997, Lueg 2001). Sveiby (1997) defines knowledge as the capability to act effectively. Leonard and Sensiper (1998) go beyond and identify knowledge as information that is relevant, actionable and based at least partially on experience. Nonaka et al. (2001) further describe knowledge as justified true belief that is rational, dynamic, humanistic and context-specific; information would become knowledge only if personal interpretation of experience, beliefs and commitments are added. While Lueg (2001) views information as a kind of preliminary stage to knowledge, Dawson (2000) argues that knowledge and information are linked together through the processes of internalization of information into personal knowledge and externalization of personal knowledge into information. Additionally, Polanyi (1962)

and Krogh et al. (2000) divide knowledge into tacit and explicit. Tacit knowledge (or know-how) that gains through individual insights overtime, is personal, complex and hard to communicate as well as formalise because it is resided in human, mind and body in terms of beliefs, assumptions, behaviours, perceptions, actions, procedures, routines, commitments, ideals, values and emotions (Goh 2002, Martensson 2000, Nonaka & Takeuchi 1995, Nonaka et al. 2001). Conversely, explicit knowledge (or know-what) is structured and relatively simple. It can be captured, recorded, documented, codified and shared using formal and systematic language in the forms of manuals, patents, reports, documents, assessments, databases, scientific formulas and other IT media (Goh 2002, Martensson 2000, Nonaka & Takeuchi 1995, Nonaka et al. 2001).

KM attempts to manage and capitalize on knowledge that accumulates in the workplace (Martensson 2000). This is achieved by organizing formal and direct process to create, store, retain, evaluate, enhance and increase knowledge for the future benefit of the organization (Dawson 2000, Smith 2001). There are slight variations among researchers in describing the process of KM. For example, Wiig (1997) divides the process into knowledge building, transforming, organizing, deploying and using, whereas Chait (1999) depicts that the KM process is based on capturing, evaluating, cleansing, storing, providing and using of knowledge. In this research, we summarize KM by dividing the entire process into five stages: create, store, make available, use and evaluate knowledge (as illustrated in Figure 2.4).

Figure 2.4 Five Stages of Knowledge Management

Nonaka et al. (2001) suggest that there are four methods to create organizational knowledge by means of interaction between explicit and tacit knowledge. The first method is socialization (Nonaka et al. 2001). It is the process of developing new tacit knowledge from tacit knowledge embedded in human or organization through experience sharing, observation and traditional apprenticeship. The second method is called externalization (Nonaka et al. 2001). This is the process of turning tacit knowledge into new explicit knowledge simply by transforming tacit knowledge in the form of document such as manual and report. The third method is combination (Nonaka et al. 2001). This is the process of merging and editing “explicit knowledge from multiple sources” into a new set of more comprehensive and systematic explicit knowledge. The last one is called internalization (Nonaka et al. 2001). This is the process of embodying explicit knowledge as tacit knowledge by learning, absorbing and integrating explicit knowledge into individual’s tacit knowledge base. The second and third stages of KM, store and make available are often linked with technologies. Explicit knowledge created is collected and stored in some sort of database or knowledge base in which the users have the right to access using “search and retrieve” tools, intranets, web access and applications, groupware and so on (Alavi & Leidner 1999, Prusak 1999, Smith 2001). Rather than reactively respond to arisen difficulties, knowledge should be used in a proactive way. Bailey and Clarke (2001) suggest that

knowledge usage could be aligned to four managerial aspects: 1) front line manager who focuses on existing performance management, is responsible to add value to current process with the aid of operational knowledge, 2) senior functional manager should make use of knowledge about functional requirements, performance expectations, resource and technical capability as well as potential to implement and coordinate organizational strategy, 3) senior executive who is positioned to develop and exploit potential strategy should leverage external knowledge to predict trends in the uncertain future, 4) technical specialist who is in charge of enhancing future and current operational performances should utilize revolutionary specialist knowledge to contribute on the processes, products, services and challenges of the particular business. Newman (1997) also emphasizes a company that can effectively exploit knowledge has the ability to deliver new market values and determine prices in the world market. The fifth stage of KM is knowledge evaluation. This phrase eliminates incorrect or outdated knowledge (Alavi & Leidner 1999). In other words, organization must keep creating new knowledge and to replace any knowledge that has become invalid (Dawson 2000).

Therefore, KM, unlike other generic solution, is capable of sustaining long term competitive advantage, but how can this be achieved? Sharkie (2003) indicates rival company still can duplicate and imitate the process of KM or even its technology, but they can never copy the knowledge and skills of employees. The spirit of KM encourages organizations to create and use knowledge continuously and also enables them to take initiative in innovating and enhancing service, product and operation. Though KM is a fundamental factor behind a company's success, several issues must be handled carefully. A minority views KM as another repackaging of IT project and even confuse KMS with IS because their concepts and functions are alike (Lueg 2001). KMS is an IT-based system designed to fit in the KM process: knowledge creation, storage/retrieval, transfer and application (Alavi & Leidner 1999). Smith (2001) clearly states that technology is only a tool used to store and disseminate knowledge but technology itself adds no value to knowledge. Goh (2002) highlights that the cooperation and collaboration among groups, individuals and leaders in knowledge transfer and sharing can add value to knowledge. Level of trust, time availability, leaders' participation, environment setting, organizational structure and

monetary as well as non-monetary rewards are keys to motivate knowledge transfer and sharing (Coulson-Thomas 1997, Goh 2002, Martensson 2000).

2.3 Software Agent

Computer program departed from the earliest stage of computer-specific application to object-oriented paradigm in which the concepts of object, abstract data type, polymorphism, inheritance and encapsulation are promoted. Object-oriented approach is very popular mainly because of its reusability, extensibility, flexibility as well as ability to construct and abstract complex system (Danforth & Tomlinson 1988). Unfortunately, the concepts and mechanisms of objects, classes and modules in object-oriented programming are insufficient to model real world complex problems due to the passive nature of object and the inflexibility of action choice within an invoked method (Jennings 2001). An innovative notion called software agent technology is developed to cope with real world complexity. The advantages of agent-oriented approach include: 1) the naturalness in modularizing components in terms of the objectives they achieve, 2) the ability to control and decide their own actions in dealing with system's inherent complexity, 3) the significant reduction in problems associated with coupling of components due to the use of high-level agent communication language, 4) the noticeable reduction in problems associated with managing relationship between software components due to the use of bottom-up inter-agent interaction (Jennings 2001).

The idea of software agent is based on Carl Hewitt's concurrent actor model that proposed the concept of a self-contained, interactive and concurrently executing object or actor in which its internal state is encapsulated and has the ability to respond to messages from other similar actors (Nwana & Ndumu 2002). Researchers and scholars still cannot concede on the definition of software agent. It is unavoidable that the argument will continue for a while until they compromise on a widely accepted interpretation. According to Lupton and Stojkovic (1998), agent is one that does things and acts on behalf of someone or something. Nienaber and Cloete (2003) and Liu et al. (1999) further elaborate software agent as a computer program that behaves

like human and is capable of autonomous actions in pursuit of specific goal. Moreover, software agent is required to function continuously, flexibly and intelligently so as to communicate, respond, determine, predict and cooperate in a particular environment without human intervention (Bradshaw 1997, WeiB et al. 2003). This is necessary especially when the problems involve multiple agents because the agents will need to interact with one another either to achieve their individual objectives or to manage the dependencies (Jennings 2001). Contrarily, Petrie (1996) disagrees with the above agreements and claims software agent is no more than a piece of ordinary application program. He also complains that most commercial agent products are just sales gimmick and have no major differences from existing technology. Rather than struggling with its definition, Nwana (1996) tends to regard software agent as an umbrella term that covers a range of more specific agent types and then continue to list and define what these other agent types are. By using “a topology of agents”, Nwana (1996) classifies existing software agents into seven categories: collaborate agents, interface agents, mobile agents, information/Internet agents, reactive agents, hybrid agents and smart agents. No doubt that there are other methods to classify software agents but it is widely accepted agents must possess at least one of the following characteristics: autonomy, reactivity, proactiveness, collaborativeness, mobility, adaptability, personality, temporal continuity, communication ability, flexibility, learning ability and intelligence.

Personality refers to the capability of manifesting the attributes of a believable character (Bradshaw 1997). The extent of personality mainly depends on who the agent frequently interacts with. If the agent has to interact with human regularly, it is beneficial to include high degree of personality in order to ensure the “smoothness” of interaction and reduce misunderstanding. Another characteristic of software agent is temporal continuity. Here, temporal continuity means the persistence of identity and state over long period of time (Bradshaw 1997). Since agent activities normally involve a sequence of actions that lasts for certain amount of time, the stability of the agent is a key to maintain the integrity of the whole process. Flexibility is the ability to choose suitable actions in proper sequence in response to the state of the external environment (Liu et al. 1999). This characteristic separates agent technology from traditional software application in a way that once a method in the traditional software

is invoked, the entire actions are performed. In contrast, the agent can decide the most appropriate action corresponds to the current situation.

Autonomy is the ability to perform its task without direct control or with only minimum supervision (Nienaber & Cloete 2003). To achieve the preset goal, autonomous agent is expected to “sense when to start”, act, response as well as make its own decision according to the environment without seeking approval from human. However, in order to guarantee the agent is under control, user must ensure the degree of autonomy is just enough for the agent to complete the task. According to Patra and Mohanty 2001), reactivity refers to the ability to perceive the environment and respond to them appropriately. Brenner et al. (1998) divide reactive agent into true reactive agent and deliberative agent. To react with the environment, the formal has suitable sensor whereas the latter possesses its own internal model of environment and from which it can draw its own conclusion. Contrarily, proactiveness is the ability to accomplish its design objective in a dynamic and unpredictable environment (Zambonelli & Wooldridge 2003). In other words, proactive agent is able to take the initiative with the intention of pursuing the predetermined goal.

Another agent characteristic is collaborativeness. It is the ability to cooperate with other agents to perform tasks in open and time-constrained multi-agent environments (Nwana & Ndumu 2002). Software programmer is impossible to code every single scenario due to the complexity of the real world but the collaborativeness characteristic allows the agent to overcome difficulties by sharing information and negotiating for specialized service with each other. Liu et al. (1999) defines mobility as the ability to transfer itself across different environment through the network. Even if the agent does not possess enough resources or required service to fulfil the goal, the ability to mobilize allows the agent to navigate across the network until it has reached the target host in which mobility agent can take advantage of the required service or resources. In spite of the advantages, Brenner et al. (1998) express their concerns on issues of security, data privacy and management. In addition, communication ability is the capability to communicate with other agents as well as human (Rykowski & Cellary 2004). Communication ability is crucial when agent's resources or ability is inadequate to remove the barrier. Under this circumstance, the

agent can communicate with other agents or human users to obtain information, resources, service or permission to tackle the problem.

Agent that possesses adaptability can adjust its behaviour according to new goals and other environment changes (Shehory & Sturm 2001). Such an action is performed automatically and fundamentally based on previous experience. Learning ability refers to the capability to learn with the purpose to improve its decision making algorithm (Talukdar 1999). Brenner et al. (1998) claim that learning ability is closely related to adaptability. For example, if agent detects new resources while some of its own resources prove to be outdated and of limited use, agent is expected to learn and adapt its behaviour accordingly. Nwana and Ndumu (2002) argue that intelligent agent should possess both the learning ability and adaptability so that the agent would have to learn and adapt as it acts or interacts with the external environment. Here, intelligence refers the extent of reasoning and learning ability to carry out the task in accordance with user's goal (Bradshaw 1997). Thus, its performance is going to increase with time.

2.4 Web-based System

The World Wide Web (WWW) has completely overshadowed other Internet applications and becomes the largest consumer of Internet backbone bandwidth (Comer 2000). The WWW is originally designed to allow people to retrieve or browse information on static web pages by clicking on the related URL but the potential of web is far from that. Nowadays, web technology is exploited to accommodate a wide variety of flexible, dynamic and interactive activities that range from simple applications, to multimedia web pages, to sophisticated business systems, to complex software applications. The rapid development of wireless network further breaks down the traditional boundary of desktop computer to be the only web-accessed device (Menkhaus 2001). Currently, mobile devices such as laptop computer, 3G mobile phone and personal digital assistant are capable of taking part in web activities (Menkhaus 2001). The above environment undoubtedly helps to accelerate the popularity of web-based system.

Kock (2002) believes that web-based system has a close relationship with the emergence of e-commerce, e-trade, e-business and other e-'s. To a certain extreme, Redouane (2002) even considers web-based system as one of the vital elements in our daily life and one could not pass a day without using it once. Web-based application has diffused into almost every single aspect within the society and the range broadly includes shopping systems (Arlitt et al. 2001), HD systems (Kock 2002), health care systems (Ruppel & Konecny 2000), business IS (Wang 2001), simulation systems (Page 1998), legal decision support systems (Stranieri et al. 2001) and education systems (Casey 1998). Although there are numerous types of web-based systems, Ardagna and Francalanci (2002) summarise the alternatives in designing web-based architectures. Firstly, developer has to choose from thin or fat client, in other words, it is a choice of whether user interface of applications is stored and executed remotely or locally (Ardagna & Francalanci 2002). Secondly, the developer has to decide the number of tiers for organizing the application within the client and server paradigms (Ardagna & Francalanci 2002). Thirdly, the developer has to determine the total number of servers with respect to computing capacity (Ardagna & Francalanci 2002). Finally, the developer has to decide how different applications or application tiers are allocated, that is, whether to allocate multiple applications on the same or separate computers (Ardagna & Francalanci 2002). No matter which alternatives the designer selects, the web-based system basically consists of web client, web server, application server and database server (Arlitt et al. 2001, Hadjerrouit 2001, Zou & Kontogiannis 2000). Web client provides user interface for the web-based system whereas web server is responsible for interacting with web client and application server (Hadjerrouit 2001). To be precise, the web server captures request from the web client and delivers the request to the application server in which the database server is utilized to support information retrieval with the purpose to prepare response for rendering on user interface (Arlitt et al. 2001).

Java is one of the major programming languages used in web-based system. Apart from its basic capabilities such as platform independence and class reusability, Java is an object-oriented programming language that possesses high degree of dynamism to allow itself to be run on different platforms or browsers without the need to be ported to a different environment or even recompiling and re-linking (Kuljis 2000).

Additional advantages include the ability to support sophisticated animation as well as smaller, cleaner, safer and easier to learn as compared to other programming languages (Kuljis 2000).

Albeit the ubiquity of web-based application, academic attention and support are definitely not sufficient, especially when dealing with System Development Life Cycle (SDLC). System analyst is always confused whether the tools and methods in traditional SDLC are still valid in developing innovative web-based system. Ruppel and Konecny (2000) clearly state that traditional SDLC is still vital in web-based development. However, supplementary concern must be addressed at user participation level so that the complexity of the system and the number of levels of users are directly proportional to the levels of user participation (Ruppel & Konecny 2000). It is also not uncommon to divide a complicated web-based system into different subsystems while separate teams are in charge of developing their own components. To ensure the quality of the final product, Redouane (2002) emphasizes the importance of direct communication between users and different teams. Thorough testing must be performed initially on each subsystem and finally on the integrated solution (Redouane 2002). Last but not least, Menkhaus (2001) suggests that the design of web-based system must cope with the relatively small and low-resolution display monitor within a diversity of mobile devices that have web accessibility.

2.5 Conclusion

The emergence of IT has converted a large part of organizational activities from manual and paper-based to automatic and electronic-based. Such a conversion not only increases the complexity of IT infrastructure, but also leads to the shift of HD's role from a traditional non-profit-making function to the one that is responsible for maintaining the optimal productivity of the organization. This conversion increases the HD's coverage on software, hardware, network and other IT related areas, which in turn results in the increase of incoming enquiries. To deal with the above issues, HD experts and researchers have continued to develop new systems, support models and support structures for HD. However, these developments are often not on the

right direction to relieve HD from tremendous amount of incoming enquiries especially when there is increasingly trend of insufficient manpower. Besides, the design and development of HD technology and support structure have made it easier for users to contact HD.

As it is impossible to design and develop systems that are free of problems and bugs, we should aim to relieve the workload of the HD by shifting some of the troubleshooting facilities to users. An approach toward this direction is therefore desirable. The literature review in this chapter shows that KM allows HD to manage and capitalize on its knowledge to help users to troubleshoot simple and routine problems that do not require specialized IT knowledge. Approaches such as externalization and combination are feasible to achieve this. Externalization allows tacit knowledge to be transformed to explicit knowledge, and combination allows explicit knowledge from multiple sources to be merged and edited into a new set of more comprehensive and systematic explicit knowledge. Furthermore, advances in software agent and web-based technology provide a more natural and dynamic approach to develop the KMS for the HD. Software agent with the characteristic to act autonomously, can be applied to free users from performing onerous tasks such as retrieving and storing knowledge from and in the knowledge database within the KMS environment. The communication capability of the software agent is another feature which can be applied to the KMS. We will discuss the application of KM techniques to HD in the following chapter.

Chapter 3 Application of Knowledge Management Techniques

This chapter discusses the application of KM techniques to create, store, make available, use and evaluate knowledge within the HD environment. A re-distributed KM framework that allows the re-distribution of simple and routine enquiries to a web-based user self-help KMS is proposed. This framework allows users to solve simple problems by retrieving the most appropriate solution from the proposed KMS. Physically, HD is made of HD support staff and technical equipment, nevertheless, the actual axis of the overall support process in HD is knowledge. When user requires technical support, this means s/he lacks sufficient IT related knowledge to carry out her/his duty. Therefore, the HD staff are responsible to help users to solve the problem by using knowledge resided in some form of repository, such as the human's brain, database or technical manual.

This chapter is organized as follows. Section 1 discusses the proposed conceptual KM framework that is used to manage tacit and explicit knowledge in HD. Further analysis and customization on the conceptual KM framework are presented in Section 2. This includes the discussion of the proposed re-distributed KM framework which provides a model to support simple and routine technical enquiries. Section 3 concludes the chapter.

3.1 Conceptual Knowledge Management Framework

When the five stages of KM together with IT are applied to manipulate technical knowledge in the HD, the combination approach proposed by Nonaka et al. (2001) works perfectly well in preserving HD's knowledge. This research proposes a conceptual KM framework to create, store, make available, use and evaluate HD knowledge (illustrated in Figure 3.1). The technical knowledge is created by both the approaches of externalization and combinations. Consider the following scenario that describes the techniques of externalization and combination. Externalization is used to

convert skills, techniques, experiences and perception from experts into explicit knowledge. Consider conducting a training session on Oracle database tools usage by the HD. Throughout the training session, HD staff must encourage users to raise questions so that the HD staff can recognize users' common difficulties and mistakes when using the software. Other than answering users' queries, the HD staff must also note down both users' questions and answers. The recorded questions and answers are a form of explicit knowledge elicited from the skills, techniques and experiences of the HD staff using the technique of externalization. On the other hand, combination is used to combine and revise explicit knowledge from manual, guidebook and training documentation into a more systematical organized knowledge. For example, the HD has organized ten training sessions on the usage of Oracle database. The ten different sets of questions and answers can be merged and edited to become a more comprehensive and systematical set of explicit knowledge using the approach of combination. In this way, both types of knowledge are converted to a form that can be stored in an electronic repository and Structure Query Language (SQL) can be applied to allow the HD staff to retrieve the required knowledge from the repository. More advanced techniques such as search engine, agent technology and artificial intelligence can also be applied to retrieve this knowledge. The retrieved knowledge is used to resolve user's problem.

The shorter product life cycle in IT also means the knowledge resides in the repository is required to be evaluated regularly in order to maintain its validity. The invalid knowledge is either renewed and stored into the repository or removed permanently from the knowledge repository. This conceptual KM framework has provided a way to manage knowledge in the HD. Undoubtedly, in order to maximize its effect, a certain degree of customization may be required depending on the organizations.

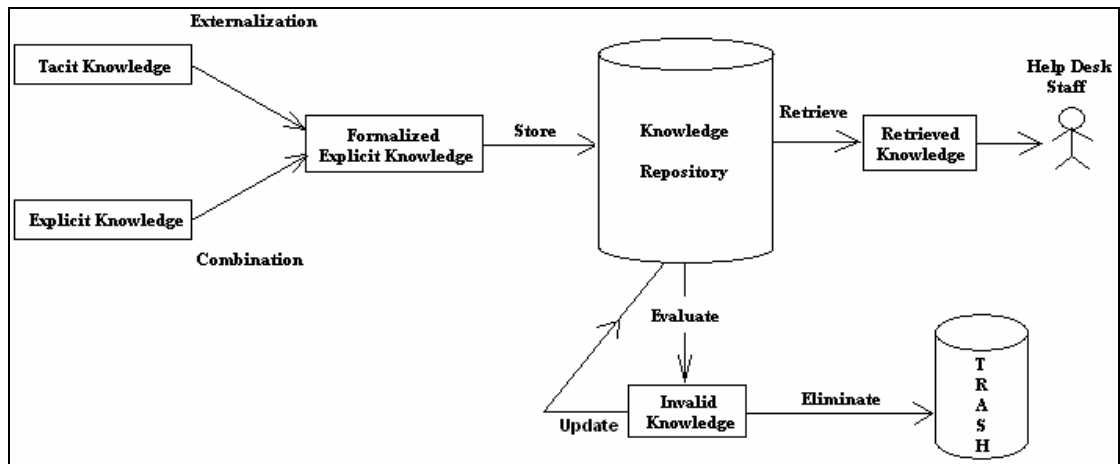


Figure 3.1 Conceptual Knowledge Management Framework

3.2 Proposed Re-distributed Knowledge Management Framework

The proposed conceptual KM framework offers an opportunity to standardize the process of managing knowledge in the HD. Both tacit and explicit knowledge are converted in a form that can be stored in knowledge base. The formalized knowledge can later be retrieved and used by the HD staff and user. No doubt the proposed conceptual KM framework enables the HD to preserve enormous amount of knowledge in a structured way, but it does not provide a way to ease the overloaded HD. The best method to ease the overloaded HD is to develop a trouble-free system, but this is technically impossible up to this moment. Since it is the enormous amount of incoming enquiries that actually cause the problem, the solution should aim to minimize incoming enquiries.

As previously discussed in Chapter 1, majority of incoming enquiries are simple and require no specialized knowledge to solve the problems (Knapp & Woch 2002 and Dawson & Lewis 2001). A more proactive approach is to reduce opportunity for user to contact HD for simple and routine technical enquiries. Instead of contacting HD, users are empowered to solve simple and routine technical problems themselves if sufficient knowledge and guidelines can be provided. We propose a framework to re-distribute simple and routine enquiries using the KM approach within to improve the support process of the HD. The proposed framework not only manages knowledge within the HD, it also has the capability to deal with the overloaded HD.

Let us first define the phrase “simple and routine technical enquiries”. Simple and routine technical enquiries in this research refer to technical problems that can be solved by user if adequate relevant information is provided without direct or indirect intervention from the HD staff. Based on the HD support areas defined by Sundrud (2002), these enquiries can be categorized into four types: IT administrative enquiries, hardware enquiries, software enquiries and miscellaneous enquiries. The IT administrative enquiries include account setup, account termination, account maintenance, account login, account suspension, password retrieval, password reset, password syntax information, password invalid, software installation and purchasing, hardware installation and purchasing as well as service purchasing. The hardware and software enquiries include performance and functional concerns in relation to various types of hardware and software. The miscellaneous enquiries include queries on missing and corrupted files, unreachable website and server plus their performances. Such categorization not only provides a structure way to further identify and elaborate simple and routine enquiries, it also helps to associate and retrieve solutions for the related enquiries. For example, software functional enquiry can be further categorized into functional enquiries of Microsoft products, Adobe products, Oracle products and so on. Thus, solutions for functional enquiries of Adobe PDF reader and Photoshop can be grouped under Adobe products category. When user has functional enquiry on Adobe products, the associated solutions of PDF Reader and Photoshop can be retrieved. Besides, the above categories may vary due to the different types of software and hardware, users, users’ skill sets and business processes.

One way to identify routine and simple enquiries is to use the reports generated by the HD management system and the automatic call distribution system. These reports provide data and information on problem type, resolution method, call duration (time required to solve the problem) and so on. By inspecting the reports in a regular manner, the HD manager can work out which enquiries are routine and simple. The proposed mechanism of identifying simple and routine enquiries is illustrated in Figure 3.2. For example, the HD management report may have indicated that there were many enquiries about “email login failure” in which most of them were related to “password invalid” and the required resolution method was merely to “reset password”. Thus by matching the above information with call duration in the

automatic call distribution system report, the HD manager could confirm the enquiries as simple and routine because the duration for each call was short. The classifications of the enquiries that have been deduced by the HD manager should be verified by the HD staff to ensure accuracy. Hence, the advice from the first level support operator is extremely important because they are in the front line answering users' enquiries daily. Therefore, they have the ability to identify simple and routine enquiries that are not found in the HD management and automatic call distribution system reports. For the purpose of this research within the IT industry, we will conduct a survey to identify a sample of simple and routine enquiries. The result of this survey will be presented in Chapter 4.

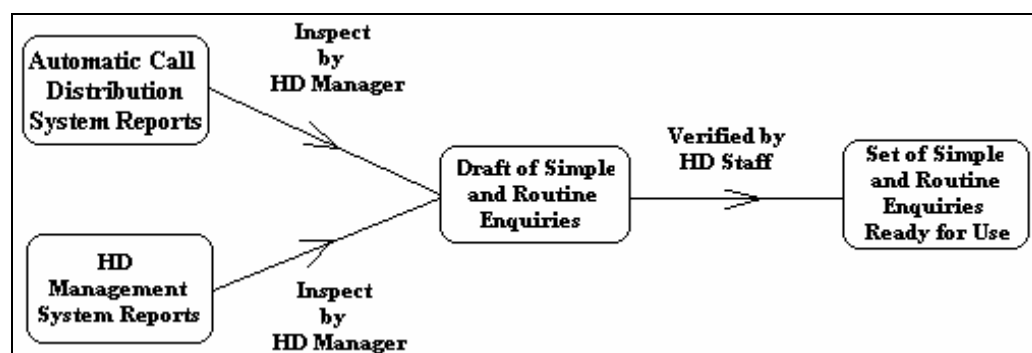


Figure 3.2 Proposed Mechanism to Identify Simple and Routine Technical Enquiries

To effectively re-distribute simple and routine technical enquiries, the proposed mechanism will be added to the proposed conceptual KM framework in Figure 3.1 and the resulting re-distributed KM model is shown in Figure 3.3. Here, rather than storing explicit knowledge into repository straight away after the processes of externalization and combination, the proposed mechanism will be applied after externalization and combination, with the aim to distinguish the knowledge and into two categories: 1) simple and routine, and 2) complex. While simple and routine knowledge is stored in a proposed web-based user self-help KMS, the complex knowledge is resided in the general knowledge repository. Consequently, users can first access the proposed web-based user self-help KMS and look for the most appropriate solution to solve their problems. Only if the solution is not available in the system, then the user can contact the HD for assistance. The repository where complex IT knowledge is resided will be used by the HD staff to answer complicated technical enquiries. Furthermore, knowledge evaluation will be conducted regularly to

remove invalid knowledge from the web-based user self-help KMS and the complex knowledge repository to ensure valid knowledge is stored and updated.

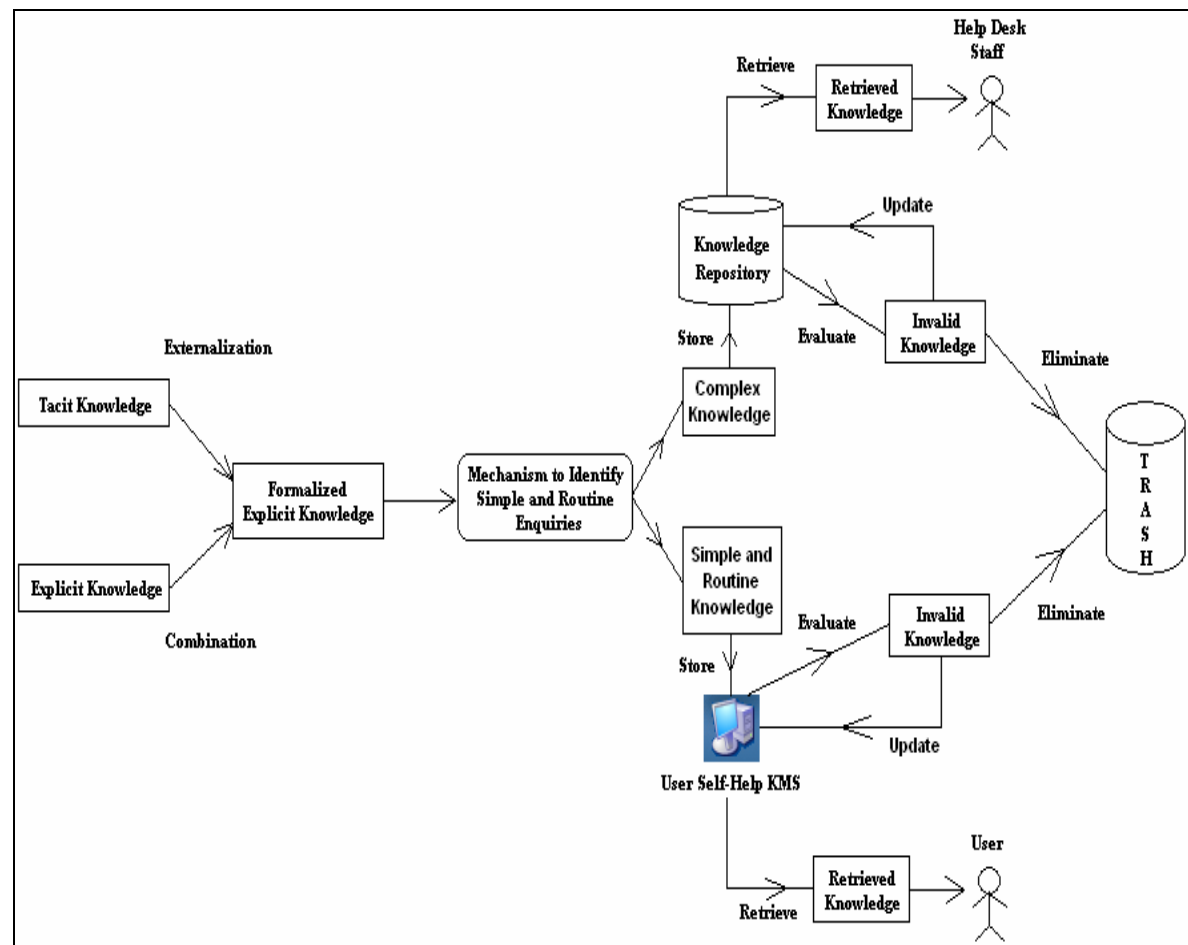


Figure 3.3 Proposed Re-distributed Knowledge Management Framework

This framework also allows the proposed web-based self-help KMS to be tailor-made in accordance with user's skill sets. For instance, if the target group of users only possesses low to medium IT skills, the KMS should avoid adopting "keyword search" as the front end user interface because the target users may find it hard to describe the problems using their own words. As IT knowledge often contains a lot of technical terms and jargons, the HD staff can rephrase and simplify the resolutions stored in the proposed system to ensure users understand the resolution methods. Figure 3.4 illustrates the basic architecture of the proposed user self-help KMS. There are five basic components within the architecture: user's browser, interface agent, search agent, resolution knowledge base which stores solutions for simple and routine technical enquiries and the interface data repository which stores information required

to facilitate user communication. Modern web technology is used as a means to deliver the system through the Internet and can appear on the browser to facilitate the interaction with the user and deliver user request for resolution. On the other hand, software agent technology is used to free user from onerous search duty by dedicating itself to look for the most suitable solution in the extensive database based on user's requirement. Moreover, it is also used to facilitate user communication. The following steps describe how the proposed web-based user self-help KMS will be deployed.

- To activate the proposed web-based user self-help KMS, the user simply clicks on the target URL. Subsequently, the interface agent that possesses communication capability will deliver a dynamic user interface to the browser, based on the information stored in the interface data repository. The dynamic and interactive communication capabilities of the interface agent help users to identify and present their problems. Firstly the interface agent interacts with the user by asking the user to select an enquiry type on the user interface. Based on the input, the interface agent will generate the next category of possible problem scenarios. This type of interaction will continue until the agent has gathered sufficient information to process the query.
- When the problem is described through the deployment of the interface agent, the search agent will be deployed to search for possible solutions. The search agent, which possesses “the ability to act autonomy”, is responsible for this task. Here, “the ability to act autonomy” refers to the capability of an agent to perform its task without direct control from the user or with only minimum supervision and direction. To achieve the preset goal of finding the most appropriate resolution, the search agent will be deployed as soon as the agent is able to “sense” that sufficient information has been gathered. The search agent will then examine the contents in the resolution knowledge base, make its own decision to select a solution according to user's problem description and finally return the solution to the user.

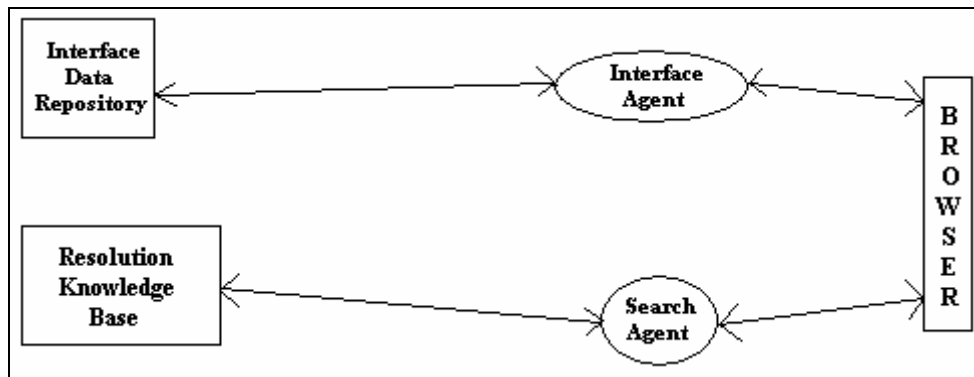


Figure 3.4 Basic Architecture of the Proposed User Self-help KMS

The proposed re-distributed KM framework not only retains the characteristic of the proposed conceptual KM framework to create, store, make available, use and evaluate knowledge, but it also helps to minimize a large amount of incoming enquiries for HD. In other words, users are able to resolve simple and routine enquiries by retrieving the most suitable solution from the proposed web-based user self-help KMS instead of using HD. To demonstrate the difference between two frameworks, let us take a look at the following example.

On one Monday morning, John is very frustrated because he cannot login to his email account with his usual password. He decides to call the HD right away. Monday morning is considered to be peak hours for the HD because quite a number of users had changed their email passwords the previous Friday and most users cannot remember the new passwords when they return to work on Monday. He waits on the phone queue for about fifteen minutes and Mary who works as the first level operator in HD, is finally available to pick up his call. Mary carefully listens to John's problem and asks him to make sure the "Num Lock" on the keyboard is on. She also reminds him to disable the "Caps Lock" on the keyboard since email password is case sensitive. Then Mary asks John to try the password again. John still cannot get into his email account and receiving the same error message "password invalid" as before. Suddenly, John remembers that he had changed his email password last Friday before he finished his work, but he is unable to remember that password now. Not wasting any time, Mary quickly walks user through to access the password reset webpage where John can reset his password to the default. Subsequently, John is successfully login to his email account using the default password. Before hanging up the phone,

Mary reminds John to change the default password because of security reason. Afterwards, Mary needs another five minutes to open a ticket, fill in troubleshooting details and close the ticket in the HD management system. Under the conceptual KM framework, John takes approximate twenty-five minutes to solve this extremely simple enquiry while Mary requires fifteen minutes to complete the whole support process.

Under the proposed re-distributed KM framework, John can access the proposed web-based user self-help KMS as soon as he realizes the email login problem. By selecting a few keywords that best describe the problem, the resolution will be delivered and displayed on the user interface of the proposed user self-help KMS within a second. Based on the resolution guidelines, John is able to login his email account with the default password. Within the proposed re-distributed KM framework, John only requires ten minutes or less to solve the same problem. On the HD side, Mary is available to perform high level and proactive support activities when John conducts his own troubleshooting task. This scenario demonstrates that the proposed re-distributed KM framework allows HD to better utilize its resources and manpower.

3.3 Conclusion

Researches have confirmed the majority of incoming enquiries belong to simple and routine enquiries (Knapp & Woch 2002 and Dawson & Lewis 2001), a KM approach is proposed to ease the workload of the HD by empowering users to resolve this type of problem. To ease HD from enormous amount of simple and routine enquiries, the proposed conceptual KM framework can be customized. The proposed re-distributed KM framework enables the simple and routine enquiries to be re-distributed to a proposed web-based user self-help KMS. It enables users to solve simple problems by retrieving the most appropriate solutions from the user self-help KMS without direct or indirect interventions from the HD.

Chapter 4 Identification of Simple and Routine Enquiries

This chapter presents the findings of the survey conducted to identify the simple and routine technical enquiries in HD. As discussed, we defined simple and routine enquiries as technical problems that can be solved by users themselves if adequate relevant information is provided without direct or indirect intervention from the HD staff.

This chapter is organized as follows. Section 1 discusses the research methodology. Profile of respondents is presented in Section 2 and Section 3 presents the identification of simple and routine incoming enquiries. The identification of incoming enquiry patterns is presented in Section 4. Discussion of the survey results is provided in Section 5 and Section 6 concludes the chapter.

4.1 Research Methodology

To identify the simple and routine enquiries, we have conducted a survey with the purpose of finding the types of simple and routine technical enquiries within the IT industry. We have sent emails to invite HDs that belong to thirty-six universities in Australia and email subscribers of the ISWorld (<http://www.isworld.org>), a mailing list of IT professionals and IS researchers and educators working in colleges and universities throughout the world, to participate in this survey. The participants were asked to respond to an online survey (<http://aroc.uow.edu.au/nelson>) which contains eighteen questions of multiple choice and short answers. Appendix A gives the questionnaire. Out of the eighteen questions, questions 1 to 6 were designed to collect data relating to the respondent's general formation, questions 7 to 11 aim to identify the classification of simple and routine technical enquiries and finally, questions 12 to 18 request information on incoming enquiry patterns.

4.2 Profile of Respondents

There were 192 logins, however only 24 usable responses. Tables 4.1 to 4.8 depict the profile of the respondents participated in the survey. The respondents are allowed to choose more than one items in survey questions 4 and 6. Over 58% of the respondents serve less than 5000 users and more than 20% of the respondents have user base of more than 20000 individuals. Three quarter (75%) of the respondents employ less than 15 staff and more than one-third of the respondents (37.5%) hire less than 5 staff. In terms of the ratio of HD staff to the users, only 16.7% of the respondents have a ratio of 1 to less-than-100 users per HD staff and more than half of the respondents (51%) have a ratio of 1 to 300-or-more users per HD staff. The ratio is calculated for each respondent, by dividing the number of individuals in the user base by the number of HD staff. Out of the total staff employed by the respondents, only 62.5% are hired as full time staff and the rest (37.5%) are made up of part time staff. Most of the respondents (62.5%) operate less than 60 hours a week and over 62% of the respondents contain more than one level in their support structure. Close to 80% of the respondents think “single point of contact” is the best term to describe their support model but only 3 respondents reported that they adopt e-support as their support model. HD management system and Internet/Web Interface are the most common tools used in the HD.

	Number of Respondents	Percentage
Less than 5000 users	14	58.3%
5000-9000 users	3	12.5%
10000-19999 users	1	4.2%
20000 users or more	5	20.8%
Unspecified	1	4.2%
Total	24	100%

Table 4.1 Help Desk User Base (Refer to Survey Question 1)

	Number of Respondents	Percentage
Less than 5 staff	9	37.5%
5-9 staff	6	25.0%
10-14 staff	3	12.5%
15-19 staff	2	8.3%
20-24 staff	1	4.2%
25 staff or more	2	8.3%
Unspecified	1	4.2%
Total	24	100%

Table 4.2 Number of Help Desk Staff (Refer to Survey Question 2)

	Number of Respondents	Percentage
Less than 1 help desk staff to 100 users	4	16.7%
1 help desk staff to 100-199 users	4	16.7%
1 help desk staff to 200-299 users	1	4.2%
1 help desk staff to 300-399 users	4	16.7%
1 help desk staff to 400-499 users	1	4.2%
1 help desk staff to 500-599 users	3	12.5%
1 help desk staff to 600 users or more	4	16.7%
Unspecified	3	12.5%
Total	24	100%

Table 4.3 Ratio of One Help Desk Staff to Number of Users

	Number of Staff	Percentage
Full time staff	135	62.5%
Part time staff	81	37.5%
Total	216	100%

Table 4.4 Distribution of Part-time and Full-time Staff (Refer to Survey Question 2)

	Number of Respondents	Percentage
Less than 40 hours	1	4.2%
40-49 hours	11	45.8%
50-59 hours	3	12.5%
60-69 hours	4	16.7%
70 hours or more	3	12.5%
Unspecified	2	8.3%
Total	24	100%

Table 4.5 Number of Operational Hours per Week (Refer to Survey Question 3)

	Number of Respondents	Percentage
Decentralized help desk	4/24	16.7%
Single point of contact	19/24	79.2%
Distributed help desk	3/24	12.5%
Outsourcing	2/24	8.3%
e-support	3/24	12.5%

Table 4.6 Help Desk Support Model (Refer to Survey Question 4)

*Respondents can select more than one answers

	Number of Respondents	Percentage
One level support	6	25%
Two levels Support	9	37.5%
Three levels Support	6	25%
Other	1	4.2%
Unspecified	2	8.3%
Total	24	100%

Table 4.7 Help Desk Support Structure (Refer to Survey Question 5)

	Number. of Help Desks	Percentage
Automatic call distributor system	3/24	12.5%
Interactive voice response system	1/24	4.2%
Help desk management system	10/24	41.7%
Expert System	1/24	4.2%
Remote Control System	6/24	25.0%
Knowledge Management System	4/24	16.7%
Internet / Web Interface	10/24	41.7%

Table 4.8 Help Desk Tools and Equipments (Refer to Survey Question 6)

*Respondents can select more than one answers

4.3 Identification of Simple and Routine Enquiries

Tables 4.9 to 4.13 depict the identification of simple and routine technical enquiries. The respondents are allowed to choose more than one items in each of the survey question 7 to 11. The majority of respondents believe if sufficient information is provided, user has the ability to resolve IT administrative issues such as password reset (58.3%), account suspension (83.3%), account login problem (62.5%), account maintenance (54.2%) and account setup (50%). The respondents also think that only hardware purchasing (54.2%), software purchasing (62.5%) and software installation (75%) guidelines should be provided to user. However, nearly all respondents think it is out of user's ability to solve any of the hardware problems. Within the software problem categories, the respondents believe user should first attempt to solve software "cannot start" (58.3%) and functionality problems (66.7%) before contacting the HD. The majority of respondents disagree that user should attempt to solve server performance (20.8%) and unreachable (16.7%) problem, website performance (29.2%) and unreachable (37.5%) problem as well as file corruption (25%) and missing problem (41.7%).

	Number of Respondents Agreed	Percentage
Account setup	12	50.0%
Account termination	6	25.0%
Account maintenance	13	54.2%
Account login problem	15	62.5%
Account suspension	20	83.3%
Password retrieval	5	20.8%
Password reset	14	58.3%
Password syntax information	7	29.2%
Password invalid	9	37.5%

Table 4.9 Administrative Issues can be Resolved by User if Sufficient Information is Provided (Refer to Survey Question 7)

*Respondents can select more than one answers

	Number of Respondents Agreed	Percentage
Hardware installation	7	29.2%
Software installation	18	75.0%
Software purchasing	15	62.5%
Hardware purchasing	13	54.2%
Service purchasing	8	33.3%
Other	2	8.3%

Table 4.10 Guidelines should be Provided to User if Needed (Refer to Survey Question 8)

*Respondents can select more than one answers

	Number of Respondents Agreed	Percentage
CD / DVD ROM	6/24	25.0%
Scanner	7/24	29.2%
Printer	12/24	50.0%
Hard drive tower	3/24	12.5%
Monitor	8/24	33.3%
Phone headset	6/24	25.0%
Mouse	10/24	41.7%
Phone handset	4/24	16.7%
Keyboard	10/24	41.7%
Other	1/24	4.2%

Table 4.11 Hardware Problems User should Attempt to Solve before Using Help Desk if Sufficient Guidelines is Provided (Refer to Survey Question 9)

*Respondents can select more than one answers

	Number of Respondents Agreed	Percentage
Software Performance	7	29.2%
Software Functionality	16	66.7%
Software can't start	14	58.3%

Table 4.12 Software Problems User should Attempt to Solve before Using Help Desk if Sufficient Guidelines is Provided (Refer to Survey Question 10)

*Respondents can select more than one answers

	Number of Respondents Agreed	Percentage
Website too slow	7	29.2%
Server too slow	5	20.8%
Website Unreachable	9	37.5%
Server Unreachable	4	16.7%
File Missing	10	41.7%
File Corruption	6	25.0%

Table 4.13 “Other” Problems Users should Attempt to Solve before Using Help Desk if Sufficient Guidelines is Provided (Refer to Survey Question 11)

*Respondents can select more than one answers

4.4 Identification of Incoming Enquiry Patterns

Tables 4.14 to 4.23 depict the identification of incoming enquiry patterns in HD. More than 66% of the respondents indicate the answers provided for questions 12 to 18 are based on estimation. Over 29% of the respondents receive less than 1500 incoming calls per month and one-fourth of the respondents (25%) receive less than 1500 incoming enquiries per month. About 38% of the respondents have experienced an increase in the incoming enquiries over the past twelve months but only 2 respondents claim a decrease in the enquiries. The reason for such an increase is summarized into “user awareness”, “staff increment” and “innovative technology”. While telephone calls (36.4%) are still the major source of contact between HD and users, the second major source of contact comes from Internet/email (20.2%). Among the three support levels in HD, first level support is the busiest since it resolves 46.9% of the incoming enquiries whereas second level and third level support only solve 13.5% and 3.8% of the enquiries respectively. The respondents also point out that hardware/software installation (15.4%), software problem (13.3%) and account/password enquiries (12.8%) currently dominate the major composition of incoming enquiries.

	Number of Respondents	Percentage
Management Report	4	16.7%
Estimation	16	66.7%
Unspecified	4	16.7%
Total	24	100%

Table 4.14 Basis of Information Provided for Question 13-18 (Refer to Survey Question 12)

	Number of Respondents	Percentage
Less than 500 calls	2	8.3%
500-999 calls	2	8.3%
1000-1499 calls	3	12.5%
1500-1999 calls	0	0%
2000-2499 calls	1	4.2%
2500 calls or more	3	12.5%
Unspecified	13	54.2%
Total	24	100%

Table 4.15 Average Number of Incoming Calls per Month (Refer to Survey Question 13)

	Number of Respondents	Percentage
Less than 500 enquiries	2	8.3%
500-999 enquiries	3	12.5%
1000-1499 enquiries	1	4.2%
1500-1999 enquiries	1	4.2%
2000-2499 enquiries	1	4.2%
2500 enquiries or more	1	4.2%
Unspecified	15	62.5%
Total	24	100%

Table 4.16 Average Number of Incoming Enquiries per Month (Refer to Survey Question 14)

	Number of Respondents	Percentage
An increase	9	37.5%
A decrease	2	8.3%
No change	6	25%
Unspecified	7	29.2%
Total	24	100%

Table 4.17 Increase / Decrease / No Change in Incoming Enquiries in the Past 12 Months (Refer to Survey Question 15)

Reasons
More awareness that help desk is available to assist
Confidence in help desk support
Greater user dependency
Greater user expectations
New hires
More users
Insufficient training to guide users to operate the systems
Greater technical complexity
Mass application / operation system update
More software, hardware and applications to support
Provision of more services
Introduction of new technology

Table 4.18 Reasons for an Increase in the Incoming Enquiries over the past 12 Months (Refer to Survey Question 15)

Reasons
Better information provided on-line
Provided more training to increase user's general IT knowledge
Provided information guidelines and technical documentations
Enhanced hardware, software and network performance

Table 4.19 Reasons for a Decrease in the Incoming Enquiries over the past 12 Months (Refer to Survey Question 15)

Reasons
No major change in IT

Table 4.20 Reasons for No Change in the Incoming Enquiries over the past 12 Months (Refer to Survey Question 15)

	Percentage
By telephone	36.4%
Walk-in	9.9%
By fax	0.2%
By Internet / Email	20.2%
Other	4.2%
Unspecified	29.2%
Total	100%

Table 4.21 Major Source of Contact (Refer to Survey Question 16)

	Percentage
Resolved by first level support	46.9%
Resolved by second level support	13.5%
Resolved by third level support	3.8%
Resolved by vendor	2.1%
Resolved by other	0.4%
Unspecified	33.3%
Total	100%

Table 4.22 Incoming Enquiries Solved by First / Second / Third Level Support (Refer to Survey Question 17)

	Percentage
Hardware / Software Installation	15.4%
Other Hardware Problem	8.3%
Other Software Problem	13.3%
Data Communication	6.1%
Voice Communication	2.5%
Account / Password	12.8%
Other	4.3%
Unspecified	37.5%
Total	100%

Table 4.23 Composition of Incoming Enquiries (Refer to Survey Question 18)

4.5 Discussion

Table 4.3 indicates the ratio of one HD staff to the number of users. The results show that 17 respondents have a high ratio of 1 to more-than-100 users per HD staff. A respondent has indicated that the HD has hired only 17 staff but it has a huge user base of 42000 individuals. This means that a single staff has to service 2471 users in this particular HD. Out of the 17 staff, only 5 are full time staff while the rest are part time staff. One can easily imagine the predicament of this HD if there is a sudden outage on one of the essential systems: more than 40000 users call at the same time but only a few are available to answer the calls. This example illustrates a possible reason for service delivery failure and user dissatisfaction in HD (Heckman and Guskey 1998). The result also demonstrates an example of unrealistic demands on HD's workload in which HD with a handful of staff has to deal with a large number of users while the staff are simultaneously expected to handle high level support issues, to participate in proactive support activities and to attend regular trainings.

The results from this survey also show that most of the HDs have experienced an increase in the incoming enquiries over the past twelve months (see Table 4.17). This finding is similar to the survey conducted by the Help Desk Institute (Broome & Streitwieser 2002). Although only 8.3% of respondents claim a decrease in the incoming enquiries, the reasons for the decrease are worth to discuss here. As shown in Table 4.19, the reasons for the decrease in the incoming enquiries are the availability of online information, training, information guidelines as well as technical documentations, and enhancement of hardware, software and network performance. Except for the hardware, software and network performance enhancement, the first three reasons are closely related to the term “self-support”. This means it is possible to decrease the amount of incoming enquiries to HD if users are given sufficient information through training, online documentation or written guidance. On the other hand, Table 4.22 shows that first level support staff handle 46.9% of the incoming enquiries. Based on the literature review in Chapter 2, most of the incoming enquiries resolved by first level support are routine and straight forward. Thus by providing users with sufficient information on hardware/software installation, “other” software problem and account/password enquiries that currently dominate the major composition of enquiries (see Table 4.23), it is possible to reduce by almost half the total incoming enquiries within the HD because of the ability of users to solve simple and routine enquiries. Table 4.7 also shows that only 12.5% of the respondents adopt e-support as their support model. By making use of modern Internet and data communication technologies such as online knowledge based system and remote control software, the adoption of e-support is possible to relieve a significant amount of workload from HD.

Although only a minority of the respondents agree to provide user with the service of purchasing and hardware installation guidelines (see Table 4.10), it is still a worthwhile effort because such an action can save a lot of work for the HD. For instance, if the user wants to purchase a dial-up Internet service so that s/he can access the Internet when s/he is away from the office. The first thing for the user is to contact the HD for this service. In most of the cases, the HD requires written approval from the department manager for purchasing of the dial-up service. Therefore, if the purchasing guideline is provided, the user will not contact the HD until s/he gets the written approval from the department manager. As illustrated in Table 4.11, most of

the respondents do not think users are capable of solving any of the hardware problems. It is not expected that the users will have the ability and knowledge to fix an unworkable monitor, but if there is a monitor troubleshooting guideline available, users can first follow the guideline to make sure the power point is switch on or the power cable and the monitor cable are connected properly. Thus, this troubleshooting guideline can save a significant amount of HD resources. In Table 4.13, the respondents again think that it is out of user's ability to solve the problems related to server, website and file. Nevertheless, it will be most useful if users can check whether they have made any typo error and spelling mistake in the URL or to clean up the cache before contacting the HD. Therefore, a simple guideline again can provide a walkthrough to perform this basic troubleshooting action.

4.6 Conclusion

The results of the survey shows that the HD staff are under enormous pressure, especially those who are working at the frontline (first level) support because incoming enquiries keep increasing while manpower is insufficient to deal with the user base. The results from the survey also demonstrate the need to ease the workload of the overloaded HD. To decrease the amount of incoming enquiries, it is recommended that HD provides users with some online information, trainings, information guidelines as well as simple technical documentations. Example of written or on-line documentations should at least include topics such as account setup, account maintenance, account login, account suspension, password reset, hardware purchasing, printer problem, software installation, software purchasing, software performance, software functionality and software "can't start".

Chapter 5 Prototype Development

This chapter discusses the prototype development of a web-based user self-help KMS based on the re-distributed KM framework presented in Chapter 3. The prototype provides an opportunity to investigate the important features and verify the concepts of the proposed framework. The prototype is a web-based system to provide access for user and HD staff, regardless of time and geographical restrictions.

The chapter is organized as follows. Section 1 discusses the design issues. Section 2 discusses the development perform. This includes the discussion of java, servlets and Java Server Pages (JSP), protégé and MySQL. Physical design of the prototype is discussed in Section 3. Section 4 presents ontology design. Section 5 discusses software agent design that includes InterfaceSoftwareAgent, SolutionRetrievalAgent and SolutionStoringAgent. Section 6 concludes the chapter.

5.1 Design Issues

The target users of the web-based self-help KMS are users with low to medium technical skill. Therefore the design of the system must be simple and user friendly. Subsequently, an easy to use dynamic user interface with interactive communication capability is proposed. The dynamic user interface allows users to present and identify the problems by choosing enquiry types and their symptoms from a series of drop boxes. Though it is quite common for the KMS to use keyword search as its front end interface, the dynamic user interface eliminates a lot of effort for novice users to use the appropriate and correct jargons to describe a problem. The solution database is an electronic repository where the resolutions of simple and routine enquiries are stored. We have also designed an administrative function interface to allow the HD staff to maintain and store the solutions of the simple and routine enquiries in the solution database. In the system, we label this interface as the admin function interface. In summary, the prototype provides three subsets of functionalities (see Figure 5.1):

1. the functionality to browse HD ontology via user and admin function interface.
2. the functionality to view problems and solutions from solution database based on results of ontology browsing via user and admin function interface.
3. the functionality to delete incorrect and to add new problems descriptions and solutions via admin function interface.

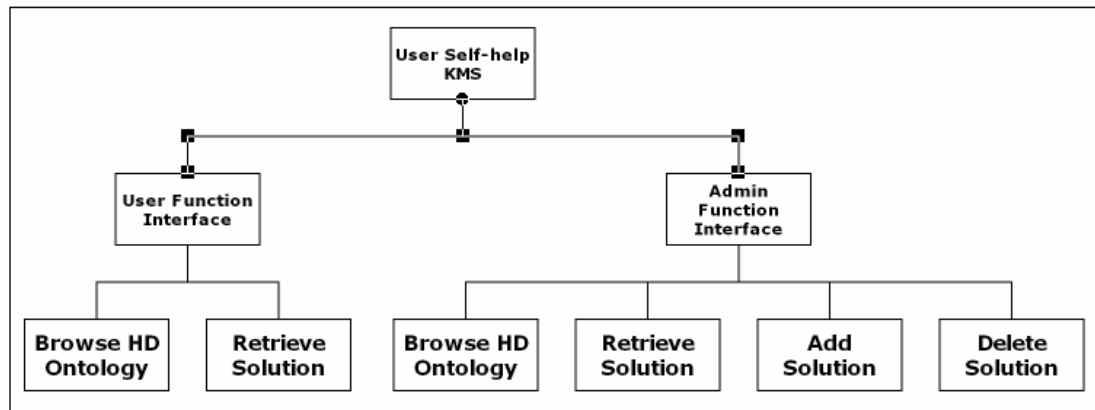


Figure 5.1 Functionalities of the Prototype

5.2 Development Platform

The proposed prototype contains seven major components to allow the required functions to perform. The seven major components are: user and admin function interface, solution database, ontology, SolutionRetrievalAgent, SolutionStoringAgent and InterfaceSoftwareAgent. The user and admin function interface are delivered through the web. This allows user and the HD staff to access the prototype by simply enter the correct URL of the prototype on their browsers. Java, a web-based programming language, is used for the development of the prototype. This includes the development of the SolutionRetrievalAgent, SolutionStoringAgent and InterfaceSoftwareAgent so that the ability to retrieve solution, store solution and execute the dynamic interface can be achieved. The advantage of using Java is its platform independent capability that allows the same java program to run on all platforms without further modification or even re-linking and recompiling. The current version of the Java 2 platform is used to prototype the system.

Since the dynamic interface cannot be displayed solely in the form of static Hypertext Markup Language (HTML), servlets and JSP are chosen to allow the interface to be displayed dynamically on the web browser. Servlets are designed to handle Hypertext Transfer Protocol (HTTP) requests and are the standard Java replacement for a variety of other methods, including Common Gateway Interface (CGI) scripts (Hall 2000). Servlets are portable between servers and operating systems (Hunter & Crawford 2001). Servlet container such as Jakarta Tomcat, is an application server that provides the facilities for running servlets. JSP extend servlet technology and allow java codes to be enclosed in special tags so that they can combine with regular HTML on the same page (Hall 2000). While the HTML is responsible to provide static content, JSP are in charge of displaying the dynamic content on the same webpage (Hall 2000). At runtime, Jakarta Tomcat, the application server, turns the JSP into a java servlet (.jsp to .java file) using a Jasper compiler. The servlet is then compiled into byte code (.class) and run on the server.

Another contribution for the dynamic interfaces is the ontology which supplies the enquiry types and symptoms within the drop boxes from a structural hierarchy of concepts. In this prototype, a java based ontology editor called Protégé is used to create the required ontology. Apart from Protégé, a java based Application Programming Interface (API) called Jena is used to allow the software agents to access the ontology. MySQL is used to provide an electronic repository for the storage of the problem description and solution. To provide connectivity between Java programs and MySQL within the prototype, MySQL Connector/J, a Java Database Connectivity (JDBC) driver, is used so that the knowledge in the solution database can be accessed by the software agents.

5.3 Physical Design of the Prototype

The prototype of the web-based user self-help KMS is accessible by users and the HD staff from regular web browser and arbitrary work station. The components of the KMS are resided in the web application directory (/webapps) of the Jakarta Tomcat

web server. JDBC driver and Jena API are required to provide connectivity between the prototype and the solution database as well as the ontology. Figure 5.2 shows the overview of the prototype's architecture.

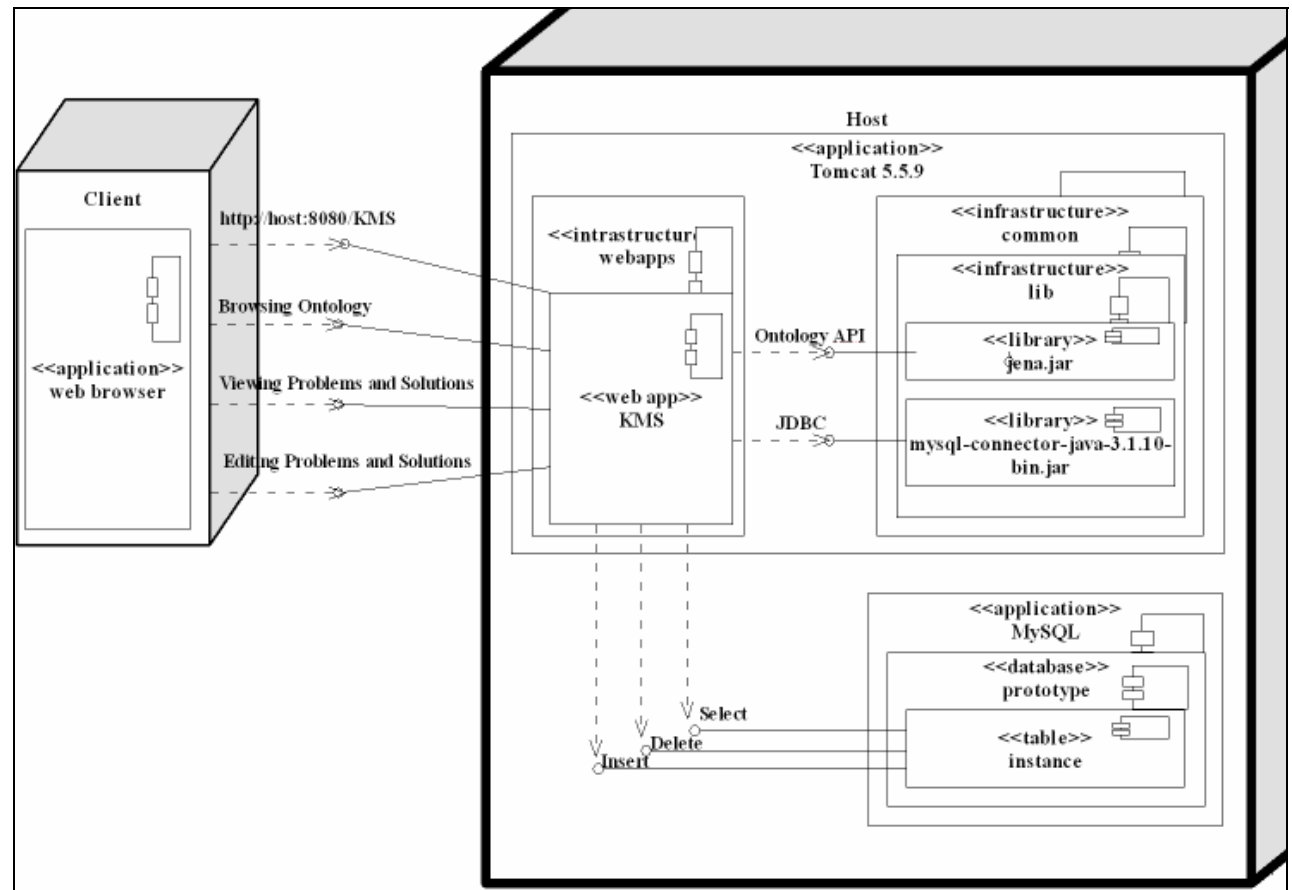


Figure 5.2 Overview of the Prototype's Architecture.

Users and HD staff can activate the prototype through the web browser simply by clicking <http://host:port/KMS> where host is the name of the machine on which the resource lives or an Internet Protocol (IP) address. Port refers to the port number on which Tomcat operates. This leads to the first interface of the prototype that consists of two entry points. The HD staff can choose the “Admin Entry” to access the admin function interface whereas users can select the “User Entry” to access the system. The admin and user entry page of the prototype is illustrated in Figure 5.3.

Figure 5.3 Admin and User Entry Page of the Prototype

Figure 5.4 shows the physical design of the dynamic interface. `DynamicInterfaceUI.jsp` allows user and the HD staff to browse and select concepts from the ontology that represents a structural hierarchy of enquiry types and symptoms. The ontology is a Web Ontology Language (OWL) based documentation that builds on the syntax of Resource Description Framework (RDF) and RDF schema. While `DynamicInterfaceUI.jsp` only provides a dynamic interface to allow users and the HD staff to browse the ontology, it is the `InterfaceSoftwareAgent` that actually reads and reasons the RDF/OWL representation of the ontology (`TechnicalSupportProblem.owl`) via Jena API.

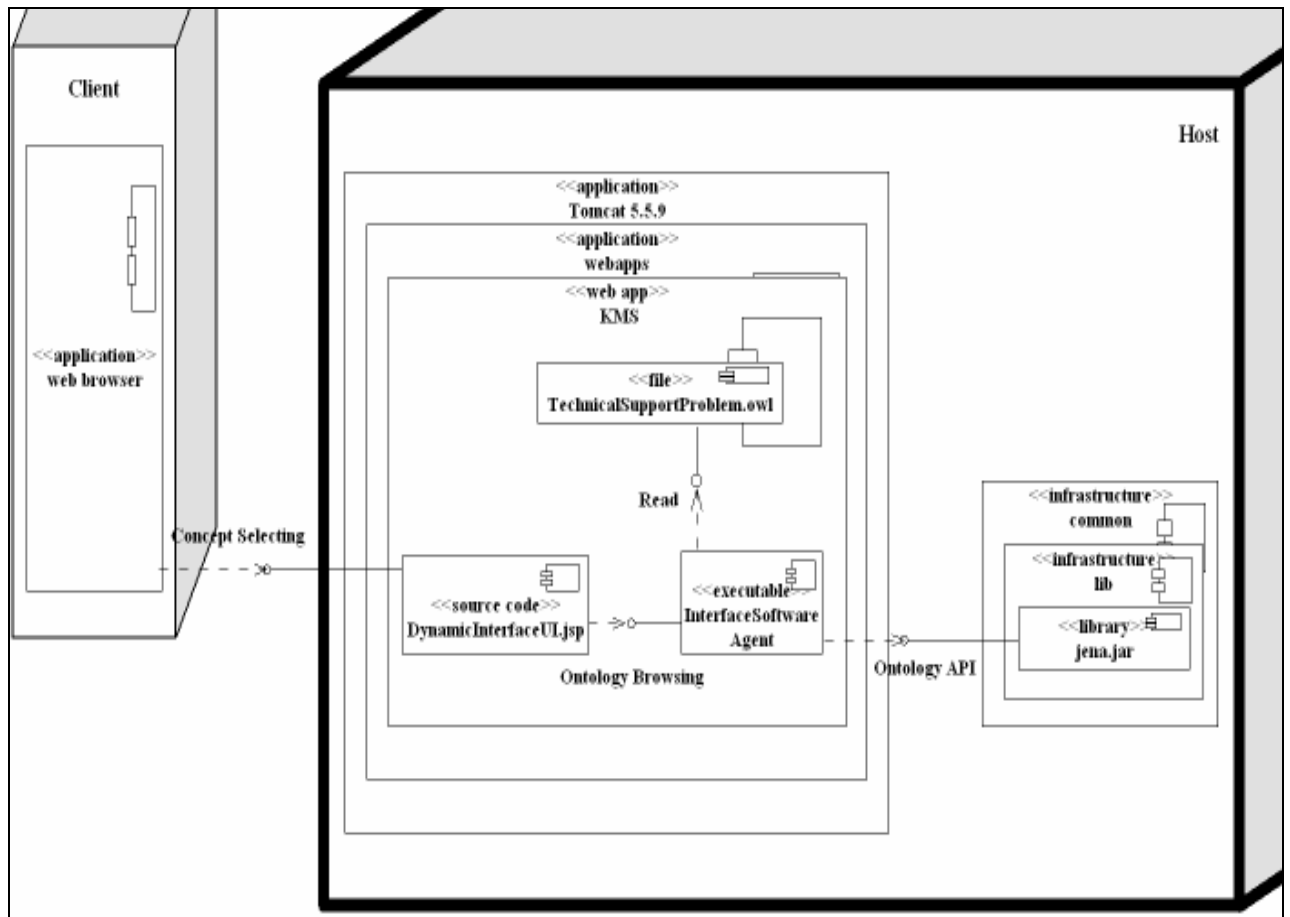


Figure 5.4 Physical Design of the Dynamic Interface

Figure 5.5 shows the physical design of the admin function interface. Other than browsing the ontology, the HD staff possess additional admin rights to insert and delete problem descriptions and solutions from the solution database through the admin function interface. The underlying technology of the admin function interface is `AdminSolutionUI.jsp`, `SolutionRetrievalAgent` and `SolutionStoringAgent`. `AdminSolutionUI.jsp` not only provides an interface for the HD staff to view and manipulate problem descriptions and resolutions based on the selection of ontological concepts, it is also responsible to forward the request of the HD staff to the `SolutionStoringAgent` and `SolutionRetrievalAgent` for further processing. While the `SolutionStoringAgent` is responsible to create new knowledge in the solution database, the `SolutionRetrievalAgent` is used to retrieve or delete the selected knowledge from the solution database. In addition, the `SolutionRetrievalAgent` is assigned of executing the refresh function for both the user and admin function interfaces which bring the interfaces to their initial state.

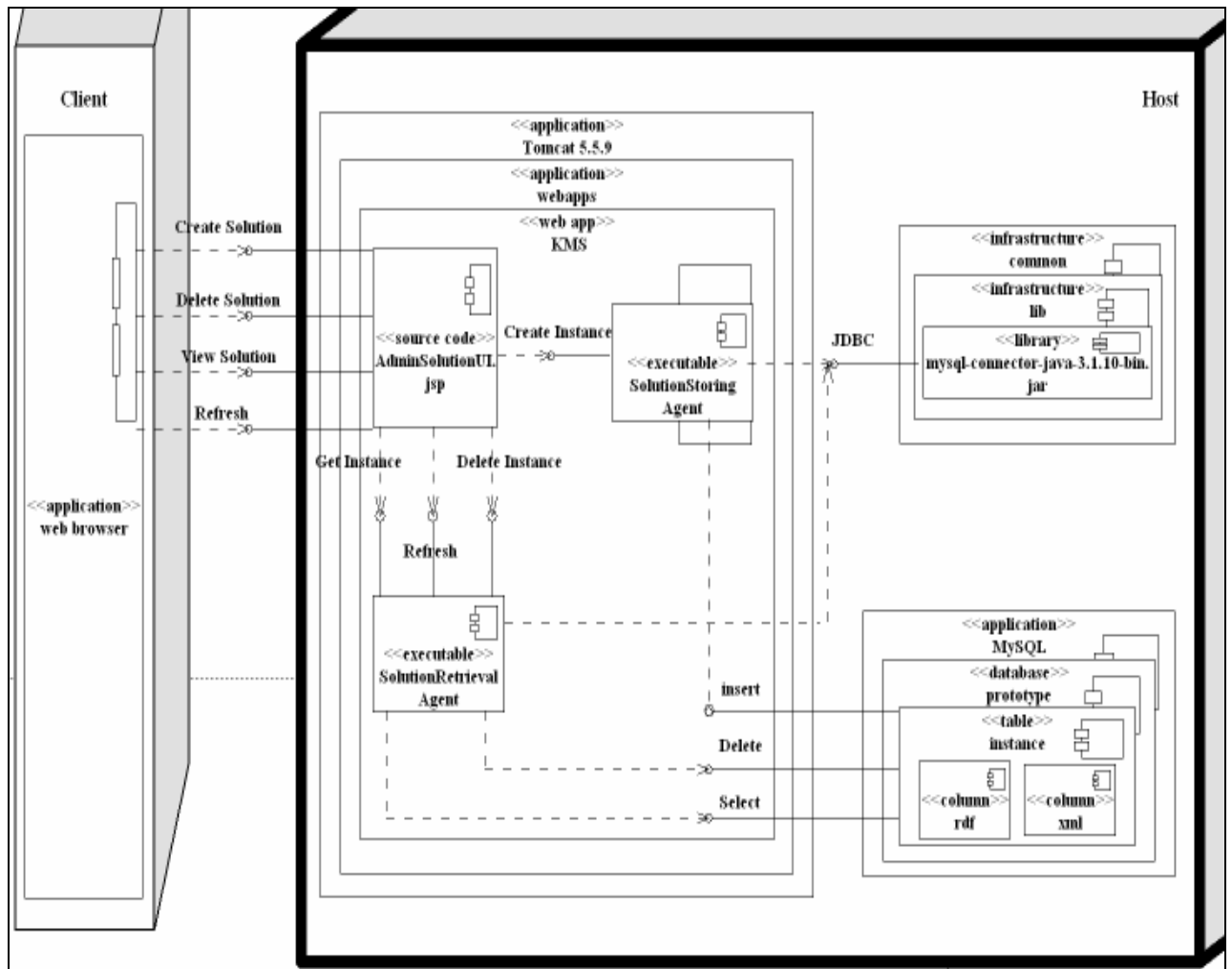


Figure 5.5 Physical Design of the Admin Function Interface

On the other hand, users have only limited access right. This limited right only allows users to view the problem description and its resolution method. Similar to the admin function interface, `UserSolutionUI.jsp` is responsible to provide an interface for displaying the problem descriptions and solutions. `UserSolutionUI.jsp` is also required to capture and send user request to the `SolutionRetrievalAgent` in order to perform the knowledge retrieval task within the solution database. The physical design of the user function interface is depicted in Figure 5.6. Appendix B shows the complete physical design of the prototype.

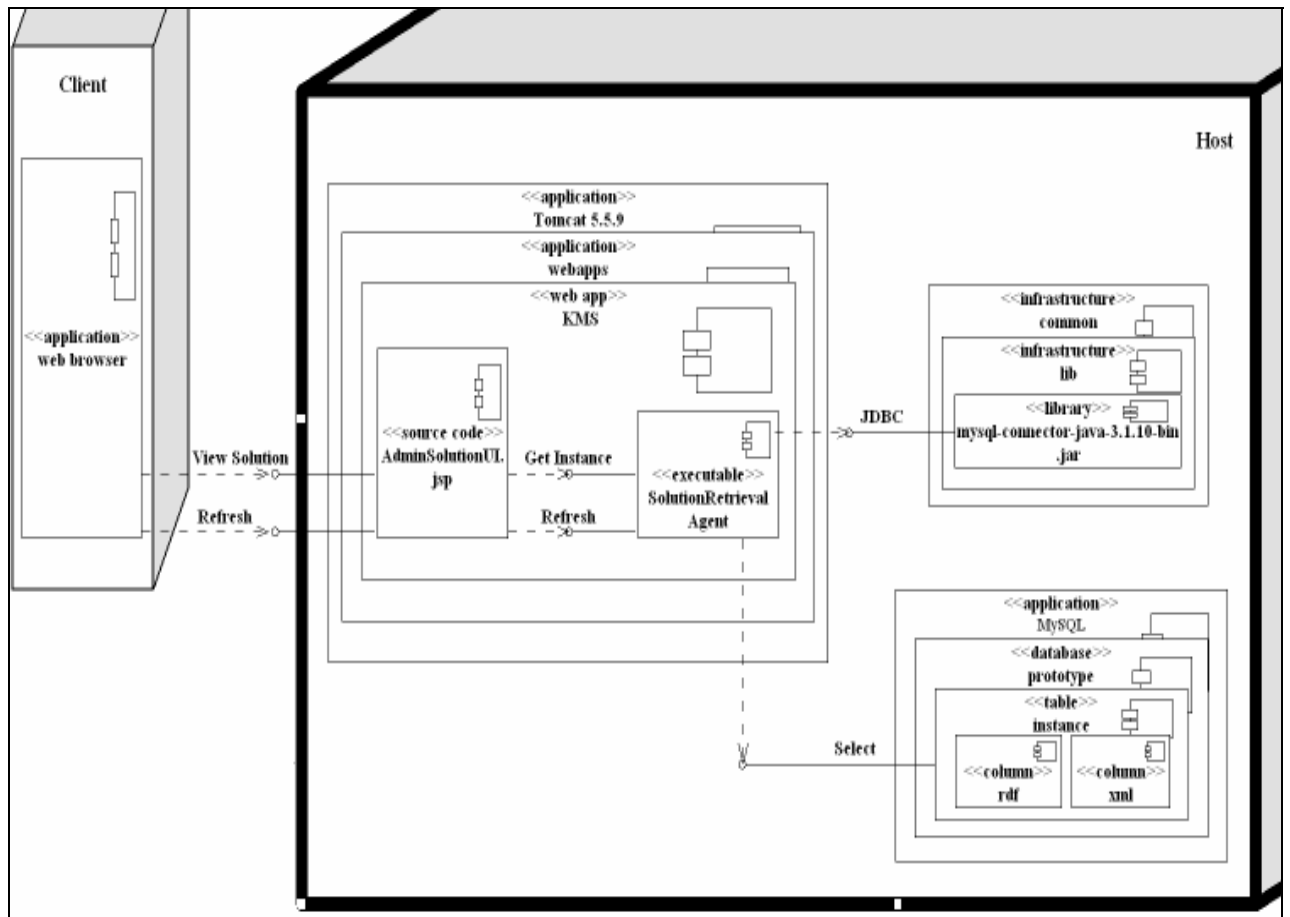


Figure 5.6 Physical Design of the User Function Interface

5.4 Ontology Design

Traditionally, the term “ontology” is defined as the study or the science of being. Gruber and Olsen (1994) first apply ontology to AI as the specifications of common conceptualizations among agents. In other words, agent is able to understand the semantic of other knowledge since knowledge is represented by the same vocabulary based on common conceptualization. The emergence of semantic web further magnifies the importance of ontology. Berners-Lee, Hendler and Lassila (2001) recognize that the HTML-based web content is solely designed for human to read and computers have no way to understand and process the semantics. In the context of the web, ontology provides a shared understanding of a domain that contains a finite list of terms and the relationships (Antoniou & Harmelen 2004). In this way, an ontology enables computer programs and software agents to understand the semantics, thus

making it possible for them to process the web content. Although different organizations may have their own ontologies, such differences can be overcome by mapping the particular terminology to a shared ontology or by defining direct mappings between ontologies (Antoniou & Harmelen 2004).

In this prototype, an OWL-based ontology is developed to represent various categories of technical enquiry types and their symptoms. The enquiry types and symptoms are used to support the dynamic interface on which users can choose to describe and identify the problems. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes, among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes (McGuinness & Harmelen 2004). The RDF uses Extensible Markup Language (XML) as interchange syntax to provide a lightweight ontology system to support the exchange of knowledge on the Web (Antoniou & Harmelen 2004, Klyne & Carroll 2004). The ontology of this prototype consists of two major categories. The first category describes the taxonomy of possible enquiry types, and the second depicts the taxonomy of symptoms in accordance with the enquiry types. Figure 5.7 depicts the enquiry types category and some of its subclasses. The enquiry types category has *Help_Desk_Enquiry* as its superclass. *Help_Desk_Enquiry* is then extended into four subclasses that include *IT_Administrative_Issue*, *Software_Problem*, *Hardware_Problem* and *Other_Problem*. These four subclasses are designed to represent the four main sources of incoming simple and routine enquiries. Further expansion of subclass and instance for each subclass is required until there is enough vocabulary to describe the problems.

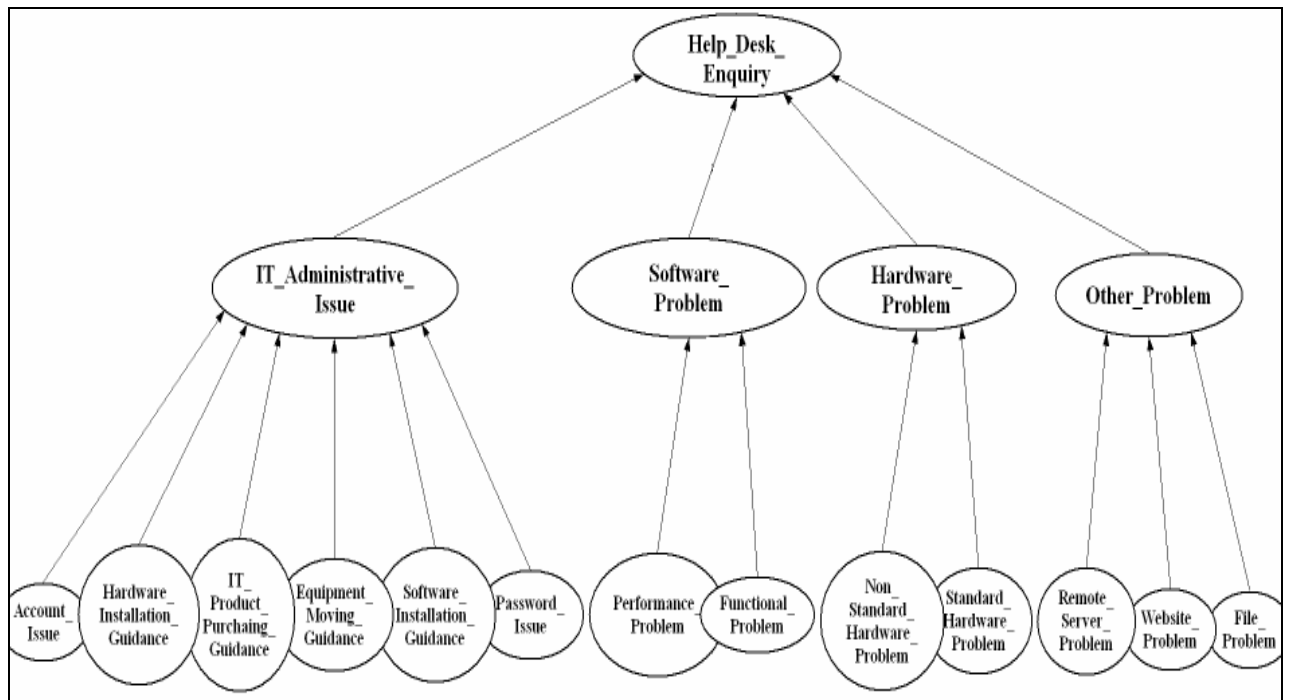


Figure 5.7 Enquiry Types Category and its Partial Subclasses

Figure 5.8 illustrates an example of the problem symptoms category and some of its subclasses. The problem symptoms class starts with *Problem_Symptoms* as its superclass. However, the expansion of this category is closely related to the enquiry types category. For example, *IT-Administrative_Issue_Symptom*, *Software_Problem_Symptom*, *Hardware_Problem_Symptom* and *Other_Problem_Symptom* are used to identify the problem symptoms of *IT_Administrative_Issue*, *Software_Problem*, *Hardware_Problem* and *Other_Problem* in the enquiry types category. The expansion of the problem symptoms category will continue until it is sufficient to identify all of the problem symptoms. Since enquiry types and problem symptoms are not standalone categories, every object in the enquiry types category are connected with an identical objects in the problem symptoms category by object properties. In OWL, object property is used to relate objects to other objects.

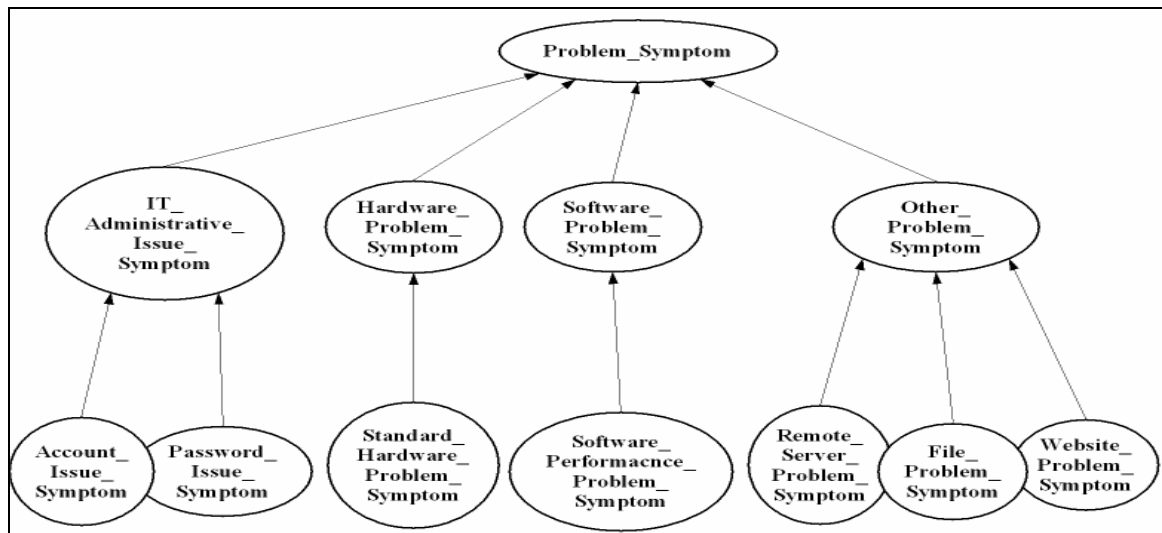


Figure 5.8 Problem Symptoms Category and its Partial Subclasses

In Figure 5.9, object property *hasFileSymptom* and its inverse, *isFileSymptomOf*, is utilized to relate *File_Problem* with *File_Problem_Symptom*. This indicates that *File_Problem* *has* *File_Symptom*, whereas *File_Problem_Symptom* is a symptom of *File_Problem*. Furthermore, the entire set of object properties and their inverses are organized in a hierarchy by using the concepts of property, subproperty and superproperty. For example, *isSymptomOf* has *isOtherProblemSymptomOf* and *isFileSymptomOf* as its subproperties. In other words, *isFileSymptomOf* has *isOtherProblemSymptomOf* and *isSymptomOf* as its superproperties. Figure 5.10 shows a partial hierarchy of the properties and their inverses.

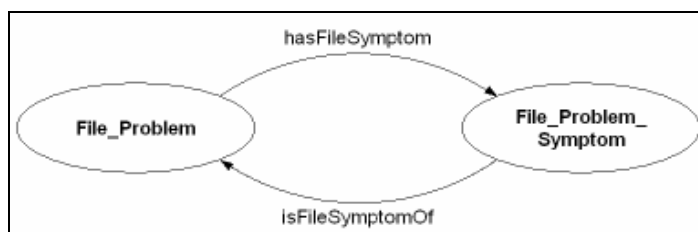


Figure 5.9 Relationships between Subclasses and Object Property (and its Inverse)

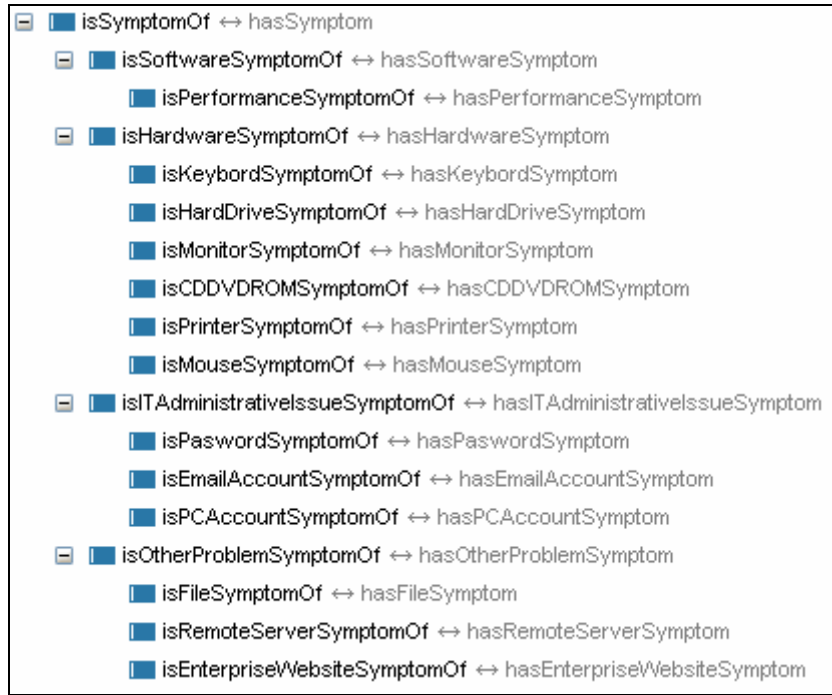


Figure 5.10 Partial Hierarchy of Properties and their Inverses

To understand how the ontology could support the dynamic interface, let us consider one branch of enquiry types and its corresponding branch of problem symptoms (see Figure 5.11). *Help_Desk_Enquiry* is the superclass of *Other_Problem* and *File_Problem*. *Other_Problem* is a subclass of *Help_Desk_Enquiry* and has *File_Problem* as its subclass. *File_Problem* is a subclass of *Other_Problem* as well as *Help_Desk_Enquiry* and it does not have any subclass. In the property hierarchy, *hasSymptom* is the superproperty of *hasOtherProblemSymptom* and *hasFileSymptom*. In term of subproperty, *hasOtherProblemSymptom* is the subproperty of *hasSymptom* and *hasFileSymptom* as its subproperty. On the other hand, *hasFileSymptom* has no subproperty, but with *hasSymptom* and *hasOtherProblemSymptom* as its superproperty. Subsequently, *File_Problem* can have instances of *File_Problem_Symptom* as values because *hasFileSymptom* and its reverse relate these two subclasses together. In this case, the instances of *File_Problem_Symptom* are *File_Corrupted*, *File_Accidentally_Deleted*, *File_Accidentally_Modified* and *Missing_File*. Besides, the concepts of the subclass, superclass, superproperty and subproperty allow *File_Problem* to inherit *hasSymptom*, *hasOtherProblemSymptom* as its own properties. The same concept also applies to *File_Problem_Symptom* that inherits *isSymptomOf* and *isOtherProblemSymptomOf* as its own properties.

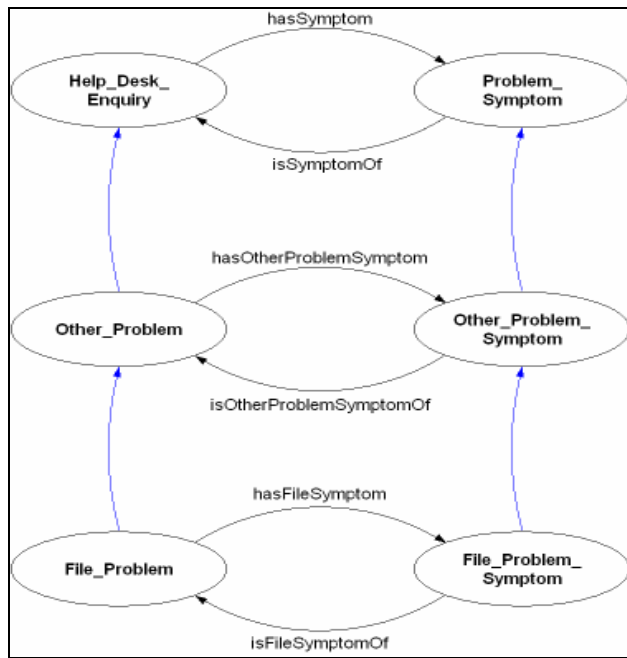


Figure 5.11 Semantic Relationships among Enquiry types, Symptoms and Properties

5.5 Software Agent Design

There are three software agents in this prototype: InterfaceSoftwareAgent, SolutionStoringAgent and SolutionRetrievalAgent. The unique characteristics in software agent technology enable the HD to customize its own user self-help KMS based on this architecture. In accordance with its own support requirements in the HD, the system can be modified by: 1) adding extra software agent, 2) removing software agent, 3) inserting additional attributes into software agent, and 4) removing existing attributes from software agent. For example, if it is decided that additional feature to allow the user to conduct an online consultation with the HD staff when users cannot find any suitable solution, then the system can add an additional communication agent that is capable of facilitating online consultation. This type of customization is straightforward and does not require major changes to the system.

The InterfaceSoftwareAgent is an agent that possesses communication capability and is in charge of providing vocabulary of enquiry types and symptoms on both the user and admin function interface, based on the concept stored in the ontology. The

vocabulary is to be used by users and the HD staff to describe the problems. Figure 5.12 shows the interaction between the InterfaceSoftwareAgent and the user.

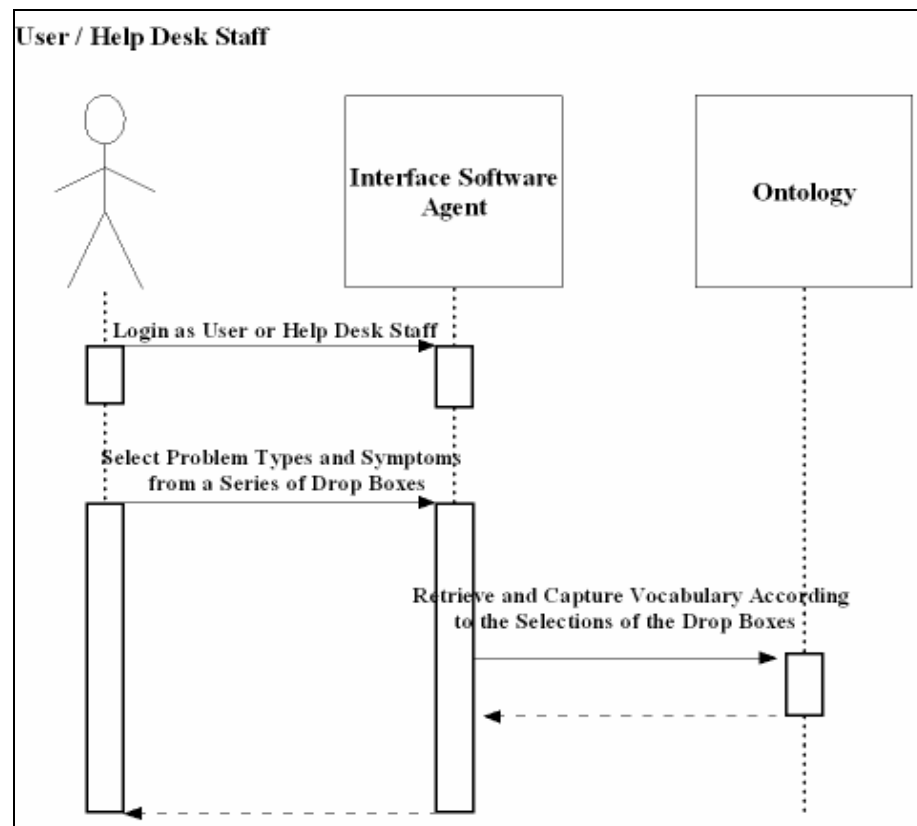


Figure 5.12 Sequence Diagram of InterfaceSoftwareAgent

When user clicks on the required link (Admin Entry or User Entry) to access the interface, it will activate the InterfaceSoftwareAgent and inform the agent whether the user is HD staff or general user. Then, the InterfaceSoftwareAgent starts to retrieve and capture vocabulary in the ontology in accordance with the selections of the drop boxes selected. The InterfaceSoftwareAgent will terminate the session if all of the vocabularies related to a particular enquiry type and symptom have been retrieved from the ontology. The retrieval and reasoning capabilities are based on a set of rules:

- 1) *Continue to capture and display all direct subclasses in the drop box, based on user's selection that relates to their superclass in the enquiry types category.*
- 2) *If there is no related subclass, capture and display all related instances from the last selected class in the enquiry types category. The InterfaceSoftwareAgent will activate the SoftwareRetrievalAgent before it terminates.*

- 3) *If there is no related subclass and instance from the last selected class in the enquiry types category, the InterfaceSoftwareAgent will examine all the object properties (includes all the inherited superproperties) that the last selected class in the enquiry types category possesses. This determines whether the direct connected class from other categories (categories other than the enquiry types) contains any instances.*
 - a) *If there is an instance in one of the direct connected class, the InterfaceSoftwareAgent will capture and display all the instances in the drop box. The InterfaceSoftwareAgent then activates the SoftwareRetrievalAgent before it terminates.*
 - b) *If there is no instance in any of the direct connected classes, the InterfaceSoftwareAgent will activate the SoftwareRetrievalAgent before it terminates.*
- 4) *If there is no related subclass, instance and object property from the last selected class in the enquiry types category, the InterfaceSoftwareAgent will activate the SoftwareRetrievalAgent before it terminates.*

Let us consider Figure 5.13 and 5.14 as an example to demonstrate the rules of the InterfaceSoftwareAgent. Using rule 1, the InterfaceSoftwareAgent starts by capturing *Hardware_Problem*, *Software_Problem*, *IT_Administative_Issue* and *Other_Problem* based on the default superclass, *Help_Desk_Enquiry*, from the enquiry types category of the ontology. The four subclasses are displayed in the first drop box. User then decides to choose *Hardware_Problem* in the first drop box. Simultaneously, the interface-software agent captures *Non_Standard_Hardware_Problem* and *Standard_Hardware_Problem* from the enquiry types category of the ontology based on user's selection in the first drop box and display these two items in the second drop box (rule 1). Subsequently, the user decides to select *Non_Standard_Hardware_Problem* in the second drop box. The InterfaceSoftwareAgent cannot find any subclass or instance related to *Non_Standard_Hardware_Problem*. Using rule 3, the InterfaceSoftwareAgent is required to gather and examine all properties, *hasSymptom*, *hasHardwareSymptom* and *isInstalledBy*, to determine if there is any direct connected classes from other category (categories other than enquiry types) contain the instances. Here, the direct connected classes are *Problem_Symptom*, *Hardware_Problem_Symptom* and

Installer. The InterfaceSoftwareAgent ignores *Problem_Symptom* and *Hardware_Problem_Symptom*, because they do not possess any property or instance. However, the InterfaceSoftwareAgent realizes that *Installer* has two instances. Thus, the two instances *Vendor* and *Help_Desk* are captured and displayed in the third drop box (rule 3a). Finally, the InterfaceSoftwareAgent activates the SolutionRetrievalAgent before terminates (rule 3a).

Figure 5.13 Example to Demonstrate the Rule of the InterfaceSoftwareAgent (Dynamic User Interface View)

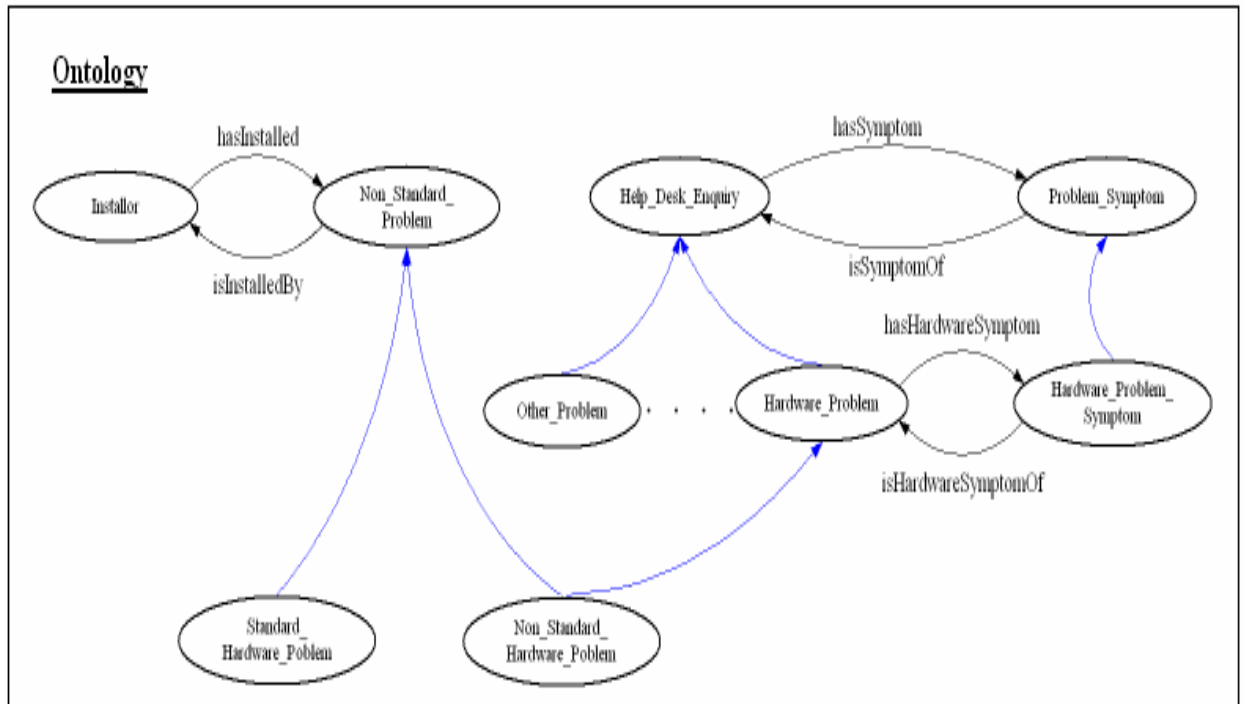


Figure 5.14 Example to Demonstrate the Rule of the InterfaceSoftwareAgent (Ontology View)

Before the InterfaceSoftwareAgent terminates, it will activate another agent called the SolutionRetrievalAgent. Here, the SolutionRetrievalAgent which possesses the ability to act autonomy is responsible to perform the solution searching task without direct control or supervision from users and the HD staff. The SolutionRetrievalAgent first gathers the OWL/RDF statements from the ontology that relate to the selected enquiry types and symptoms. By matching the OWL/RDF statements with the entries stored in the RDF column of the solution database, the SolutionRetrievalAgent can decide whether the required solution is available in the database. If there is a solution, the SolutionRetrievalAgent will transform the solution from XML to HTML format. The transformed solution will be displayed on either the user or admin function interface. If the user has logged in as general users, they can read and follow the instructions of the retrieved solutions to troubleshoot their technical problems. On the other hand, for the HD staff, the SolutionRetrievalAgent will grant the admin right to manipulate the knowledge in the solution database. In other words, the HD staff has the option to delete the solution that they just retrieved (RDF and XML entry) from the solution database. However, if the SolutionRetrievalAgent cannot find a suitable solution, a short message will be sent to the users to indicate that there is no solution currently

available for this enquiry and they should contact HD for further assistance. Contrarily, the SolutionRetrievalAgent will not send message to those who login as HD staff, if it cannot locate an appropriate solution. Instead, the SolutionRetrievalAgent will activate another software agent called SolutionStoringAgent before terminates to allow a new solution to be entered and stored in the solution database. Figure 5.15 shows the interaction among HD staff, the SoftwareRetrievalAgent and the InterfaceSoftwareAgent.

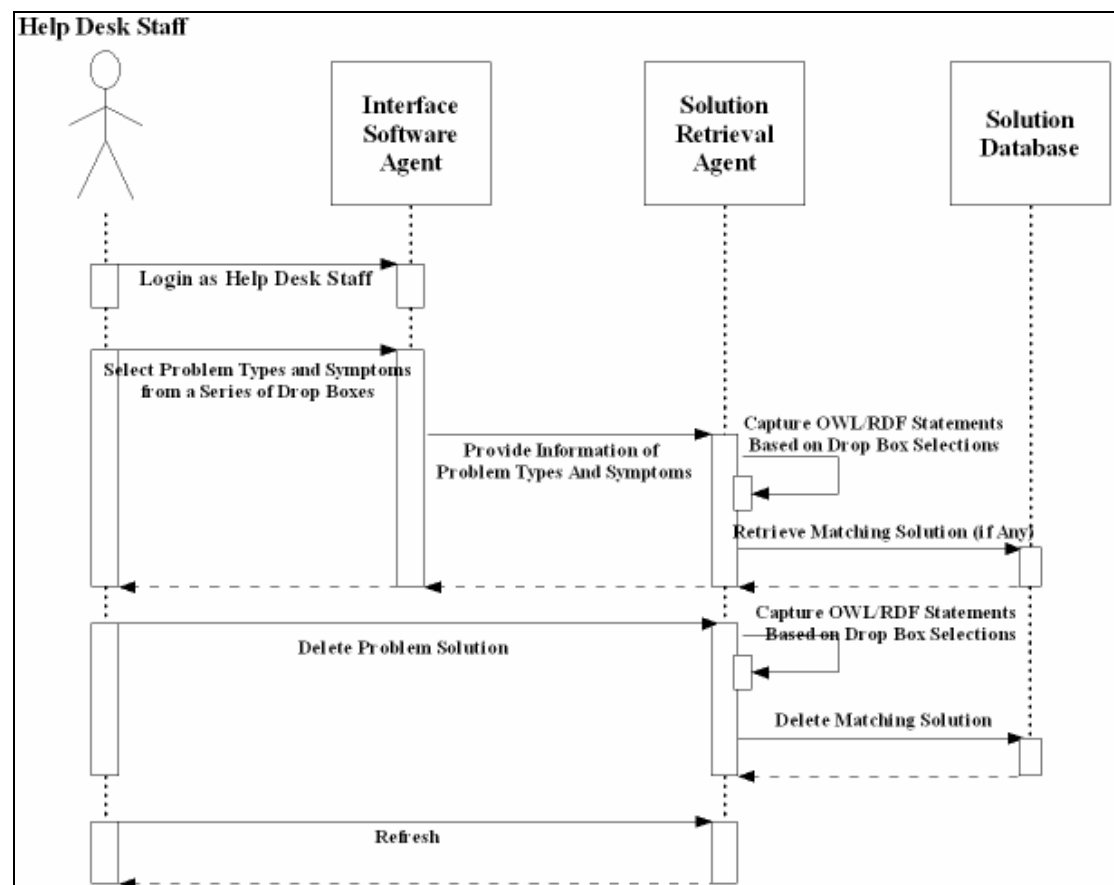


Figure 5.15 Sequence Diagram of SolutionRetrievalAgent and InterfaceSoftwareAgent

The SolutionStoringAgent is another autonomous agent in the prototype that allows HD staff to store the problem description and solution in the solution database. To perform this, the SolutionStoringAgent has to gather the OWL/RDF statements from the ontology based on the enquiry types and symptoms selected by the HD staff. Secondly, the SolutionStoringAgent has to collect the problem description and solution entered by the HD staff on the admin function interface. Thirdly, the

SolutionStoringAgent will transform the problem description and solution from HTML to XML format. Finally, the OWL/RDF statements and the XML documents will be inserted into the RDF and XML columns of the solution database and the SolutionStoringAgent will terminate afterwards. Figure 5.16 shows the activity diagram of the SolutionStoringAgent.

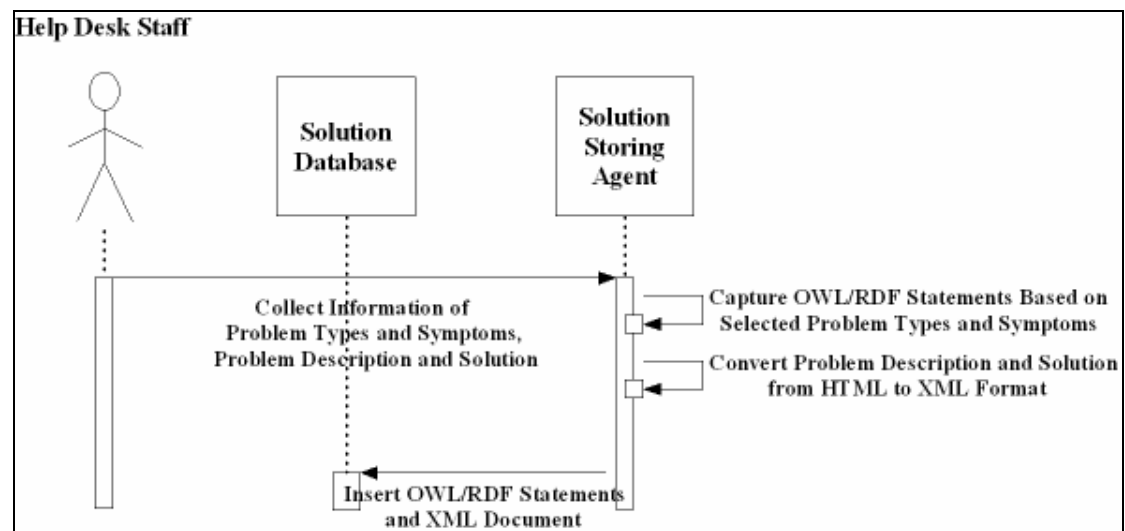


Figure 5.16 Sequence Diagram of SolutionStoringAgent

5.6 Conclusions

This chapter describes the prototype development for the web-based user self-help KMS. The java, servlets and JSP, ontology and software agents are used to allow simple and routine enquiries to be created, deleted and retrieved in the prototype. While servlets and JSP are responsible for the layout of the dynamic, admin and user interface, it is the InterfaceSoftwareAgent, SolutionRetrievalAgent and SolutionStoringAgent that actually query and reason the ontology, retrieve and store the solutions for the simple and routine enquiries. These functionalities also allow the web-based user self-help KMS to be integrated with the proposed re-distributed KM framework. Thus, users can solve their simple and routine problems by retrieving the most appropriate solution from the user self-help KMS without intervention from HD staff.

Chapter 6 Prototype Illustrations

This chapter demonstrates the prototype that has been developed in Chapter 5. It aims to demonstrate the functionalities of the prototype to store, view, delete and retrieve solutions for simple and routine technical enquiries. The prototype has two interfaces: admin and user function interface. While the admin function interface allows the HD staff to store, view and delete solutions, general users can use the user function interface to retrieve the most suitable solution for their simple and routine enquiries.

This chapter is organized as follows. The illustrations of functionalities of admin and user function interfaces are presented in Section 1. Section 2 concludes the chapter.

6.1 Illustrations of the Prototype

The prototype illustration is divided into two parts. The first part is to illustrate the admin function interface. This includes the illustration of its capability to store, view and delete solution for simple and routine technical enquiry. The second part is to illustrate the user function interface that includes the demonstration of its capability to retrieve solution for simple and routine enquiry.

6.1.1 Admin Function Interface Illustration

The admin function interface of the prototype is designed to allow the HD staff to perform the system administrative duty. To access the admin function interface, the HD staff are required to click on the “Admin Entry” button on the entry page of the prototype (see Figure 5.3). As admin users, the HD staff have the admin right to store, view, and delete solutions for simple and routine technical enquiries from the solution database. To do so, the HD staff are required to describe and identify the enquiry types and their symptoms by selecting the related vocabularies from a series of drop boxes on the admin function interface. As described in Chapter 5, the enquiry types

and their symptoms are represented by a series of interrelated vocabularies organized in a structural hierarchy within the ontology. The concepts of class, superclass, subclass, property, superproperty and subproperty in the hierarchy enable the vocabularies to form seventy different sets of incoming enquiries and their related symptoms. The complete list of seventy sets of enquiries and symptoms are shown in Appendix C.

After selecting the vocabularies for the enquiry types and their symptoms, the system will check to see if there is any matching solution that has already been stored in the solution database based on the selections of the drop boxes. If there is no matching solution in the database, the prototype will bring up two text fields (see Figure 6.1). The HD staff can then enter the problem description in the first text field and problem solution in the second text field. Once the “OK” button is pressed, the problem description and solution will be saved in the solution database. On the other hand, if the prototype has found a matching solution in the database, the solution will be retrieved and displayed. To delete the solution, the HD staff can click on the “Delete” button and it will be removed from the solution database permanently.

Figure 6.1 Admin Function Interface

To illustrate the functionalities of the admin function interface, let us consider two scenarios. In the first scenario, the HD staff is required to create a solution for “equipment moving enquiry” that belongs to the IT administrative issue. To do so, the HD staff is required to choose *IT_Administrative_Issues* and *Equipment_Moving_Guidance* in the first and second drop boxes. Since there is no matching solution found, it will bring up the “Problem” and “Solution” text fields as illustrated in Figure 6.2. Then, the HD staff has to enter the problem description in the “Problem” text field and the problem solution in the “Solution” text field. Then, the HD staff can click on the “OK” button to save this new problem and solution (see Figure 6.3).

Figure 6.2 First Sample Screen of
Storing “Equipment Moving
Guidelines” Solution

Figure 6.3 Second Sample Screen of
Storing “Equipment
Moving Guidelines”
Solution

In the second scenario, the HD staff is required to delete the solution for “equipment moving enquiry” because the solution is no longer valid. Firstly, the HD staff is required to choose *IT_Administrative_Issues* and *Equipment_Moving_Guidance* in the first and second drop boxes. Concurrently, the solution for *Equipment_Moving_Guidance* is retrieved and displayed because this solution has already been stored in the solution database. To permanently remove the solution from the solution database, the HD staff will click on the “Delete” button (see Figure 6.4).

Figure 6.4 Sample Screen of Deleting Solution

6.1.2 User Function Interface Illustration

The user function interface of the prototype is designed to allow users to retrieve the most appropriate solutions for their enquiries. The users will login as general users by clicking on “User Entry” button. In this case, their access rights are limited to retrieve solutions only. To view the enquiry solution, user is required to describe the enquiry types and their symptoms by choosing the related vocabularies from a series of drop

boxes on the user function interface. If there is a solution for the enquiry types, it will be displayed on the interface. Otherwise, a message will be shown to inform user that the solution for the chosen enquiry types and symptoms is currently unavailable.

To illustrate the functionalities of the user function interface, let us consider two scenarios. In the first scenario, John gets an error message when he tries to access an internal website. He decides to search for solution in the web-based user self-help KMS. Firstly, he describes and identifies the enquiry types and symptoms by selecting *Other_Problem*, *Website_Problem*, *Enterprise_Website_Problem* and *Website_Error_Message* in four of the drop boxes. The prototype immediately retrieves the matching solution from the solution database and displays the solution (see Figure 6.5).

Figure 6.5 Sample Screen of Retrieving Solution

In the second scenario, John has difficulties in using some of functions in SmartDraw installed by the vendor. Smartdraw is considered as a non-standard software in the company that he is currently working for. Thus, he decides to access the web-based user self-help KMS and search for a suitable solution. John identifies and describes the problem types and symptoms by selecting *Software_Problem*, *Functional_Problem*, *Non_Standard_Software_Problem* and *Vendor* in four of the drop boxes. As there is no matching solution stored in the solution database, a message is displayed to inform John that the solution is not available and he is asked to contact the HD for assistance (see Figure 6.6).

Figure 6.6 Sample Screen of Displaying “Knowledge Unavailable” Message

6.2 Conclusion

This chapter illustrates the functionalities of the prototype, to store, view, delete and retrieve knowledge. The functionalities of the prototype allow the HD staff and users to perform KM techniques to store, make available, use and evaluate knowledge. The admin function interface allows the HD staff to perform knowledge storing technique by saving solutions in the solution database. In addition, the view and delete functions on the admin function interface also allow the HD staff to execute knowledge evaluating technique. Invalid knowledge is removed from the solution database. The retrieve function on the user function interface enables users and the KMS to perform knowledge using and making available techniques. As a result, users can use the displayed solutions retrieved from the web-based user self-help KMS to solve their simple and routine enquiries. This chapter has also demonstrated the functionalities of the software agents. Without direct intervention from the HD staff, the system is able to provide troubleshooting function via the deployment of software agents.

Chapter 7 Conclusion

This chapter concludes the presentation of the thesis. The chapter is organized as follows. Section 1 discusses research results. Section 2 outlines research contribution. The conclusion and future direction are given in Section 3.

7.1 Research Result

The aim of this research is to investigate the feasibility of developing a web-based user self-help KMS to improve the support process for routine and simple technical enquires in HD. We have developed a prototype of a web-based user self-help KMS that integrates KM techniques and software agent technology. The application of KM techniques in the prototype allows solution for simple and routine enquiry to be stored, made available, used and evaluated while the software agents are developed to execute the dynamic interface, the solution retrieval and storing tasks. Users can solve their simple and routine problems by retrieving the most appropriate solution from the system.

We have applied the KM techniques to create, store, make available, use and evaluate HD knowledge in the proposed conceptual KM framework. The re-distributed KM framework provides a way to re-route simple and routine enquiries to the proposed web-based user self-help KMS.

A survey has been conducted to identify the classification of simple and routine enquiries. The results from the survey have identified a sample of simple and routine enquiry. The results also indicate that a decrease in the amount of incoming enquires can be expected if online information, trainings, information guidelines, technical documentations and troubleshooting guidelines are provided to the users.

In this research, we have applied software agent technology in the development of the dynamic user interface in the proposed user self-help KMS. A software agent with the

ability to communicate is designed to provide common and formalize vocabularies from the ontology. The ontology enables user to describe and identify their enquiries and the related symptoms easily. An autonomous software agent is developed to retrieve the most appropriate solution from the knowledge database based on user's requirement without further intervention from HD staff.

7.2 Research Contribution

Academic researchers and HD practitioners have invested substantial resources in developing new HD models, support structures and technologies to ease the overloaded HD, however the results have not been encouraging. Most of the researches are focused on design issue that can provide users a more convenient way to contact HD. In fact, it actually encourages more users to contact the HD. To effectively relieve the overloaded HD, the solution should be focused on call flow re-distribution. In this research, we have proposed to find a way to distribute the overwhelming simple and routine enquiries. The proposed re-distributed KM framework developed in this research has demonstrated simple and routine enquiries can be re-routed to a web-based user self-help KMS. It allows users to solve their simple and routine problems without contacting the HD. This "self help" practice provides a way to ease the workload of the HD. The research has also demonstrated that advancement in software agent ontology and web-based system can be applied.

7.3 Future Research

Gruber and Olsen (1994) first applied ontology to AI so that agent is able to understand the semantic of knowledge based on the common conceptualization. Agents are able to reuse knowledge from ontologies created by other companies, departments, groups or individuals. However, the lack of standardization hinders communication and collaboration between agents because the concepts used for a particular subject can be described by different ontologies (Wiesman & Roos 2004). To reuse ontology, Pinto and Martins (2001) suggests two different methods: merging

and integration. Here, ontology merging is the process of building one ontology in one subject reusing two or more different ontologies on that subject, whereas ontology integration is the process of building an ontology in one subject reusing one or more ontologies in different subjects. Ontology merging and integration are two interesting research areas for the HD industry. The popularity of using ontology to manage technical knowledge makes it possible for HD to reuse other HDs or IT companies' knowledge in terms of ontology. Hence, the choice of ontology merging or integration is very important. For example, company A has reached an agreement with Microsoft and Adobe to allow the HD of company A to reuse technical support knowledge of Microsoft and Adobe products. This means that users and HD staff in company A can make use of Microsoft and Adobe's technical support knowledge to troubleshoot their own problems. Since company A, Microsoft and Adobe have their own ontologies, ontology integration or merging should be carried out to allow software agent from company A to retrieve technical knowledge from other companies. Further research on effective approaches that allow different ontology to merge and integrate particularly with reference to HD industry is recommended.

References

- Abraham, D. M., Spangler, W. E. and May, J. H. (1991) "Expertech: Issues in the Design and Development of an Intelligent Help Desk System." *Expert Systems with Applications*. 2(4). pp.305-319.
- Alavi, M. and Leidner, D. E. (1999) "Knowledge Management Systems: Issues, Challenges, and Benefits." *Communications of the Association for Information Systems*. Volume 1, Article 7. Feb. pp.1-37.
- Andress, M. (2001) Internal SLAs benefit the entire company. InfoWorld. 4 April. <<http://www.itworld.com/Man/2679/IWD010430tcintersla/pfindex.html>> Accessed 22 March, 2005
- Antoniou, G. and Harmelen, F. (2004) *A Semantic Web Primer*. The MIT Press. London, England.
- Ardagna, D. and Francalanci, C. (2002) A Cost-oriented Methodology for the Design of Web Based IT Architectures, *In Proceedings of the 2002 ACM symposium on Applied computing*. pp.1127-1133. Madrid, Spain.
- Arlitt, M., Krishnamurthy, D. and Rolia, J. (2001) "Characterizing the Scalability of a Large Web-based Shopping System." *ACM Transactions on Internet Technology*. Volume 1, Number 1. August. pp.44-69.
- Bailey, C. and Clarke, M. (2001) "Managing Knowledge for Personal and Organisational Benefit." *Journal of Knowledge Management*. Volume 5, Number 1, pp.58-67.
- Berners-Lee, T., Hendler, J. and Lassila, O. (2001) "The Semantic Web." *Scientific American*. <<http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2>> Accessed 5 May, 2005
- Bornhoft, M., Day, B. and Curnow, P. (1991) Computer-Switch Telephony Applications, *In Proceedings of the 3rd IEEE conference on Telecommunications*. 17-20 March. Edinburgh, UK.
- Bradshaw, J. M. (1997) "An Introduction to Software Agents." *Software Agents*. AAAI Press/The MIT Press. pp.3-16.
- Brenner, W., Zarnekow, R. and Wittig, H. (1998) *Intelligent Software Agents Foundations and Applications*. Springer.
- Broome, C. and Streitwieser, J. (2002) "What is E-support." *Service and Support Handbook*. Help Desk Institute. pp.31-40.
- Casey, D. (1998) Learning "From" or "Through" the Web: Models of Web Based Education, *In Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education: Changing the delivery of computer science education*. August 18-21. pp.51-54. Dublin City Univ., Ireland.
- Central Computer and Telecommunications Agency (1989) *IT Infrastructure Library: Help Desk*, HMSO Publication Centre. pp.1-4.
- Chait, L. P. (1999) "Creating a Successful Knowledge Management System." *Journal of Business Strategy*. March/April. pp.23-26. <<http://www.chaitassociates.com/index.html>> Accessed: 18 April, 2005
- Chase, R. L. (1997) "The Knowledge-Based Organization: An International Survey." *Journal of Knowledge Management*. Volume 1, Number 1, pp.38-49.

- Childe S., Maull, R. and Bennett, J. (2001) "Frameworks for Understanding Business Process Re-engineering." *Understanding Business Processes*. Routledge. pp.195-207.
- Comer, D. E. (2000) *Internetworking with TCP/IP Principles, Protocols, and Architectures*. Fourth Edition. Volume 1. Prentice Hall International, Inc.
- Coulson-Thomas, C. J. (1997) "The Future of the Organization: Selected Knowledge Management Issues." *Journal of Knowledge Management*. Volume 1, Number 1, pp.15-26.
- Cruess, A. (2002) Transforming a Help Desk from Average to Excellent, *In Proceedings of the 30th annual ACM SIGUCCS conference on User services*. 20-23 Nov. Providence, Rhode Island, USA.
- Czegel, B. (1998) *Running an Effective Help Desk*. John Wiley & Sons, Inc.
- Czegel, B. (1999) *Help Desk Practitioner's Handbook*. Wiley.
- Danforth, S. and Tomlinson, C. (1988) "Type Theories and Object-Oriented Programming." *ACM Computing Surveys*. Volume 20, Number 1. March.
- Dash, J. (2000) "Help Desk Outsourcing Rises." *ComputerWorld*. 26 June. <<http://www.computerworld.com/managementtopics/outsourcing/story/0,10801,46290,00.html>> Accessed: 22 March, 2005
- Dawson, E. and Lewis T. (2001) Deakin University ITS Help Desk: Co-operative Partnership as the Solution, *Presented at the 2001 Australasia Educause Conference on the Power of 3*. 20-23 May. Gold Coast, Queensland, Australia.
- Dawson, R. (2000) "Knowledge Capabilities as the Focus of Organisational Development and Strategy." *Journal of Knowledge Management*. Volume 4, Number 4, pp.320-327.
- Faulks, J. (2004) Outsource This! Broaden Support and Reduce Staff Burnout, *In Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services*. 10-13 Oct. Baltimore, MD, USA.
- Giarratano, J. and Riley, G. (1998) *Expert Systems: Principles and Programming* (3rd Edition). PWS Publishing, Boston.
- Goh, S. C. (2002) "Managing Effective Knowledge Transfer: An Integrative Framework and Some Practice Implications." *Journal of Knowledge Management*. Volume 6, Number 1, pp.23-30.
- Goker, M. and Roth-Berghofer, T. (1999) Development and Utilization of a Case-Based Help-Desk Support System in a Corporate Environment, *In Proceeding of the 3rd ICCBR International Conference on Case-Based reasoning*. 27-30 July. Monastery Seon, Munich, Germany.
- Goker, M., Roth-Berghofer, T., Bergmann, R., Pantleon, T., Traphoner, R., Wess, S. and Wilke, W. (1998) The Development of Homer: A Case-Based CAD/CAM Help-Desk Support Tool, *In Proceedings of the 4th EWCBR European Workshop on Case-Based Reasoning*. August. Dublin, Ireland.
- Grajek, S., Dannheim, G., Euben, J., Paolillo, J. and Stagg, D. (2002) Telecom and IT: Moving toward Unified Support, *In Proceedings of the 30th annual ACM SIGUCCS conference on User services*. 20-23 Nov. Providence, Rhode Island, USA.
- Greenberg, R. (1998) "Help on Help." *CIO Magazine*. 15 May. <<http://www.cio.com/archive/051598/help.html>> Accessed: 15 March, 2005.
- Gruber, T. and Olsen, G. (1994) An Ontology for Engineering Mathematics, *In Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Gustav Streemann, Bonn, Germany.

- Gurbaxani, V. (1996) "The New World of Information Technology Outsourcing." *Communications of the ACM*. Volume 39, Issue 7. July. pp.45-46.
- Hadjerrouit, S. (2001) "Web-based Application Development: A Software Engineering Approach." *ACM SIGCSE Bulletin*. Volume 33, Issue 2. pp.31-34.
- Hall, M. (2000) Core Servlets and Java Server Pages. Sun Microsystems Press.
- Hammer, M. (1997) "Re-engineering Work: Don't Automate." Obliterate. *Harvard Business Review*. July-August. pp.104-112.
- Hathaway, J. (1995) Service Level Agreements: Keeping a Rein on Expectations, *In Proceedings of the 23rd annual ACM SIGUCCS conference on User services: winning the networking game*. St. Louis, Missouri, United States.
- Heckman, R. and Guskey, A. (1998) "Sources of Customer Satisfaction and Dissatisfaction with Information Technology Help Desks." *Journal of Market Focused Managem.* Number 3. pp.59-89.
- Hunter, J. and Crawford, W. (2001) Java Servlet Programming. O'Reilly Media.
- Ives, W., Torrey B., Gordon, C. and Andersen Consulting (1998) "Knowledge Management: An Emerging Discipline with a Long History." *Journal of Knowledge Management*. Volume 1, Number 4, pp.269-274.
- Jennings, N. R. (2001) "An Agent-based Approach for Building Complex Software Systems." *Communications of the ACM*. Volume 44, Issue 4, pp.35-41. April.
- Kajiko-Mattsson, M. (2003) Infrastructures of Virtual IT Enterprises, *In Proceeding of the 2003 IEEE International conference on Software Maintenance*. 22-26 Sept. Los Alamitos, CA, USA.
- Kendall, H. (2002) "Prehistoric Help Desk!!" *Support World*. Help Desk Institute. Oct-Nov. pp.6-8.
- Ketler, K. and Willems, J. (1999) A Study of the Outsourcing Decision: Preliminary Results, *In Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research*. pp.182-189. New Orleans, Louisiana, United States.
- Kirchmeyer, R. (2002) The Consolidated Help Desk, *In Proceedings of the 30th annual ACM SIGUCCS conference on User services*. 20-23 Nov. Providence, Rhode Island, USA.
- Knapp, M. and Woch, J. (2002) Towards a Natural Language Driven Automated Help Desk, *In Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*. pp.96-105.
- Kock, N. (2002) "Managing with Web-based IT in Mind." *Communications of the ACM*. Volume 45, Issue 5. May. pp.102-106.
- Kolawa, A. (2004) "Outsourcing Devising a Game Plan." *ACM Queue*. Vol. 2, No.8. November.
<<http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=235>>
Accessed: 22 March, 2005
- Kraemer, K. L., Gurbaxani, A. and Dunkle, D. (2000) Performance Benchmarks for Information Systems in Corporations. University of California, Irvine, Graduate School of Management, Centre for Research on Information Technology and Organizations (CRITO).
- Krishna, S., Sahay, S. and Walsham, G. (2004) "Managing Cross-Culture Issues in Global Software Outsourcing." *Communications of the ACM*. Volume 47, Issue 4. May. pp.62-66.
- Krogh, G. V., Ichijo, K. and Nonaka, I. (2000) Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation. Oxford University Press.

- Kuljis, J. (2000) A Review of Web-based Simulation: Whither We Wander? *In Proceedings of the 32nd conference on Winter simulation*. pp.1872-1881. Orlando, Florida.
- Kundtz, J. M. (1996) Implementing Problem Management Processes at the Helpdesk Using the Business Process Method, *In Proceedings of the Quest for Quality and Productivity in Health Conference*. 29 Sept-2 Oct. Williamsburg, Virginia, USA.
<<http://kundtz.com/speaking/04-shspaper96/shspaper.html>> Accessed: 3 March, 2005.
- Klyne, G. and Carroll, J. (2004) Resource Description Framework (RDF) Concepts and Abstract Syntax.
<<http://www.w3.org/TR/rdf-concepts/>> Accessed 10 May, 2005
- Lee, J., Huynh, M., Kwok, R. and Pi, S. (2003) "IT Outsourcing Evolution – Past, Present and Future." *Communications of the ACM*. Volume 46, Issue 5. May. pp.84-89.
- Leonard, D. and Sensiper, S. (1998) "The role of tacit knowledge in group innovation", *California Management Review*, Volume 40, Number 3, pp.112-132.
- Liu, H, Zeng, G. and Lin, Z. (1999) "A Construction Approach for Software Agents Using Components." *ACM SIGSOFT Software Engineering Notes*. Volume 24, Issue 3, pp.76-79. May.
- Lueg, C. (2001) "Information, Knowledge, and Networked Minds." *Journal of Knowledge Management*. Volume 5, Number 2, pp.151-159.
- Lupton, W. and Stojkovic, V. (1998) "Solving incomplete and incorrect information problems using conditional planning, execution monitoring, and situated planning agents." *ACM SIGAda Ada Letters*. Volume XVIII, Issue 5, pp.87-96. Sept/Oct.
- Marcella, R. and Middleton, I. (1996) The Role of the Help Desk in the Strategic Management of Information Systems. *OCLC Systems and Services*. 12(4). pp. 4-19.
<<http://www.auditnet.org/docs/roles.pdf>> Accessed: 3 March, 2005.
- Martensson, M. (2000) "A Critical Review of Knowledge Management as a Management Tool." *Journal of Knowledge Management*. Volume 4, Number 3, pp.204-206.
- McGuinness, D. and Harmelen, F. (2004) OWL Web Ontology Language Overview.
<<http://www.w3.org/TR/owl-features/>> Accessed 5 May, 2005
- McKoen, P. (2000) Creating a Help Center from Scratch: A Recipe for Success, *In Proceedings of the 28th annual ACM SIGUCCS conference on User services: Building the future*. 29 Oct.-1 Nov. Richmond, Virginia, United States.
- McLay, A. (2003) Understanding Help Desks: A Hawke's Bay Study, *In Proceedings of the 16th Annual NACCQ, on Computing Qualifications*. 6-9 July. Palmerston, New Zealand
- Menkhaus, G. (2001) Architecture for Client-independent Web-based Applications, *In Proceedings of the Technology of Object-Oriented Languages and Systems*. 12-14 March. pp.32.
- Middleton, I. (1999) "The Evolution of the IT Help Desk: From Crisis Centre to Business Manager in the Public and Private Sectors." MSC Thesis. The Robert Gordon University, Faculty of Management, School of Information and Media. Apr. Aberdeen, UK.

- Nam, K., Rajagopalan, S., Rao, H. R. and Chaudhury, A. (1996) "A Two Level Investigation of Information System Outsourcing." *Communications of the ACM*. Volume 39, Issue 7. July. pp.27-28.
- Newman, V. (1997) "Redefining Knowledge Management to Deliver Competitive Advantage." *Journal of Knowledge Management*. Volume 1, Number 2, pp.123-128.
- Niedzwiecki, R. and Peterson, M. (2002) Help Desk Support: To Be or Not To Be, *In Proceedings of the 30th annual ACM SIGUCCS conference on User services*. 20-23 Nov. Providence, Rhode Island, USA.
- Nienaber, R. and Cloete, E. (2003) A Software Agent Framework for the Support of Software Project Management, *In Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*. pp.16-23.
- Nonaka, I. and Takeuchi, H. (1995) *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press.
- Nonaka, I., Toyama, R. and Konno, N. (2001) "SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation." *Managing Industrial Knowledge Creation, Transfer and Utilization*. Sage Publications. pp.13-43.
- Nwana, H. S. (1996) "Software Agents: An Overview." *Knowledge Engineering Review*. 11(3), 1-40. September.
- Nwana, H. S. and Ndumu, D. T. (2002) "A Brief Introduction to Software Agent Technology." *Agent Technology Foundations, Applications, and Markets*. pp.29-47. Springer.
- Oza, N., Hall, T., Rainer, A. and Grey, S. (2004) Critical Factors in Software Outsourcing – A Pilot Study, *In Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*. 5 Nov. pp.67-71. Newport Beach, CA, USA.
- Page, E. H. (1998) The Rise of Web-based Simulation: Implications for the High Level Architecture, *In Proceedings of the 30th conference on Winter simulation*. pp.1663–1668. Washington, D.C., United States.
- Patra, M. R. and Mohanty, H. (2001) A Formal Framework to Build Software Agents, *In Proceedings of 2001 Eighth APSEC Asia-Pacific Software Engineering Conference*. 4-7 December. pp.119-126. Macau, China.
- Peters, S. L. (1993) Expanding the Help Desk to Simplify Customer Access to CIT Services: The Integration of Services across Platforms, Applications and Units, *In Proceedings of the 21st annual ACM SIGUCCS conference on User services*. Nov. New York, NY, USA.
- Petrie, C. J. (1996) "Agent-Based Engineering, the Web, and Intelligence." *IEEE Expert: Intelligent Systems and their Applications*. Volume 11, Issue 6, pp.24-29. Decemeber.
- Pinto, H. and Martins, J. (2001) A Methodology for Ontology Integration, *In Proceedings of the 1st international conference on Knowledge capture K-CAP '01*. 22-23 Oct. pp.131-138. Victoria, British Columbia, Canada.
- Polanyi, M. (1962) *Personal Knowledge towards a Post-Critical Philosophy*. Routledge & Kegan Paul.
- Prescott, J., Franklin, G., Lovgren, T., Kilty, T., Cleary, A. and Mai, A. (2001) Technical Session: Evolution of Three Help Desks, *In Proceedings of the 29th annual ACM SIGUCCS conference on User services*. 17-20 Oct. Portland, Oregon, USA.

- Prusak, L. (1999) "The Nature of Knowledge and its Management." *The Knowledge Management Yearbook 1999-2000*. Butterworth-Heinemann.
- Rea, D. and Cleary, K. (2001) See for Yourself: Implementing Web-Based Remote Observation/Control, *In Proceedings of the 29th annual ACM SIGUCCS conference on User services*. 17-20 Oct. Portland, Oregon, USA.
- Redouane, A. (2002) Guidelines for Improving the Development of Web based Applications, *In Proceedings Fourth International Workshop on WSE Web Site Evolution*. 2-10 October. Montreal, Canada.
- Rob, P. and Coronel, C. (2002) Database Systems Design, Implementation and Management. Fifth Edition. *Course Technology*.
- Robson, W. (1997) Strategic Management and Information Systems. Pitman.
- Ruppel, C. and Konecny, J. (2000) The Role of IS Personnel in Web-Based Systems Development: The Case of a Health Care Organization, *In Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research*. pp.130-135. Chicago, Illinois, United States.
- Rykowski, J. and Cellary, W. (2004) Virtual Web Services – Application of Software Agents to Personalization of Web Services, *In Proceedings of the 6th international conference on Electronic commerce*. pp.409-418. Delft, The Netherlands.
- Scullen, J. (2001) Re-engineering Desktop Support at Griffith University, *Presented at the 2001 Australasia Educause Conference on the Power of 3*. 20-23 May. Gold Coast, Queensland, Australia.
- Sharkie, R. (2003) "Knowledge Creation and its Place in the Development of Sustainable Competitive Advantage." *Journal of Knowledge Management*. Volume 7, Number 1, pp.20-31.
- Shehory, O. and Sturm, A. (2001) Evaluation of Modeling Techniques for Agent-Based Systems, *In Proceedings of the fifth international conference on Autonomous agents*. 11-23 February. pp.624-631. Montreal, Quebec, Canada.
- Smith, A. (2004) "Motorway Design Must Learn from Past Mistake." *The Sydney Morning Herald*. 24 August. pp.4.
- Smith, A. (2005) "Traffic Levels Far Outstrip Predictions." *The Sydney Morning Herald*. 12 March. pp.13.
- Smith, C. L. (1996) Building a Help Desk From Scratch, With No Staff, No Equipment and No Money: Moulding Novice Student Consultants into Seasoned Help Desk Operators, *In Proceedings of the 24th annual ACM SIGUCCS Conference on User Services*. Sept. Chicago, Illinois, United States.
- Smith, E. A. (2001) "The Role of Tacit and Explicit Knowledge in the Workplace." *Journal of Knowledge Management*. Volume 5, Number 4, pp.311-321.
- Stranieri, A., Yearwood, J. and Zeleznikow, J. (2001) Tools for World Wide Web Based Legal Decision Support Systems, *In Proceedings of the 8th international conference on Artificial intelligence and law*. pp.206–214. St. Louis, Missouri, United States.
- Sundrud, R. (2002) "Computer Help Desk Services: A Case Study of Three Community Colleges in Pennsylvania." Doctor of Education Thesis. College of Education, Tempe University, Arizona, USA.
- Sveiby, K. E. (1997) The New Organizational Wealth: Managing & Measuring Knowledge-Based Assets. Berrett-Koehler Publishers, Inc.
- Talukdar, S. (1999) The Next Software Revolution, *In Proceedings of the IEEE Power Engineering Society Summer Meeting*. 18-22 July. Volume 2, pp.843-845. Edmonton, Alberta. Canada.

- Tischler, F and Trachtenberg, D. (1998) "The Emergency of the distributed help desk." *Telemarketing and Call Center Solutions*. June.
<http://www.findarticles.com/p/articles/mi_qa3700/is_199806/ai_n8801492>
Accessed: 15 March, 2005.
- Tourniaire, F. and Farrell, R. (1998) *The Art of Software Support*. Prentice Hall.
- Turban, E. and Aronson, J. E (2001) *Decision Support Systems and Intelligent Systems*. Prentice Hall. New Jersey.
- Underwood, J., Hegdahl, D. and Gimbel, J. (2003) To Corral Support, a Proper Set of Tools are Needed, *In Proceedings of the 31st annual ACM SIGUCCS conference on User services*. 21-24 Sept. San Antonio, TX, USA.
- Wang, S. (2001) "Toward a General Model for Web-based Information Systems." *International Journal of Information Management*. Volume 21. pp.385-396.
- WeiB, G., Rovatsos, M. and Nickles, M. (2003) Capturing Agent Autonomy in Roles and XML, *In Proceedings of the 2003 Second International Joint Conference on AAMAS Autonomous Agents and Multiagent Systems*. 14-18 July. pp.14-18. Melbourne, Australia.
- Whitten, J. L., Bentley, L. D. and Dittman, K. C. (2001) *Systems Analysis and Design Methods*. Fifth Edition. McGraw Hill.
- Wiesman, F. and Roos N. (2004) Domain Independent Learning of Ontology mappings, *In Proceedings of the Third International Joint Conference on AAMAS Autonomous Agents and Multiagent Systems*. 19-23 July. pp.846-853. New York, USA.
- Wiig, K. M. (1997) "Knowledge Management: An Introduction and Perspective." *Journal of Knowledge Management*. Volume 1, Number 1, pp.6-14.
- Workman, M. and Bommer, W. (2004) Redesigning Computer Call Center Work: a Longitudinal Field Experiment. *J. Organiz. Behav.* 25. Wiley InterScience. pp.317-337.
<<http://www.interscience.wiley.com>> Accessed: 3 March, 2005
- Yule, G. (1996) *The Study of Language*. Cambridge University Press. New York.
- Zambonelli, F. and Wooldridge, M. (2003) "Developing Multiagent Systems: The Gaia Methodology." *ACM Transactions on Software Engineering and Methodology (TOSEM)*. Volume 12, Issue 3, pp.317-370. July.
- Zou, Y. and Kontogiannis, K. (2000) Web-based Specification and Integration of Legacy Services, *In Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*. pp.17. Mississauga, Ontario, Canada.

Appendix A – Information Technology Help Desk Survey

Re-designing Help Desk's Support Process using Knowledge Management Framework

Research Questionnaire

Introduction

The purpose of this survey is to examine current Help Desks' support processes and to investigate processes that allow the Help Desk to provide support services in effective and efficient manners.

This survey should take approximately 10-15 minutes to complete. Your participation is voluntary and it does not report to me any personally identifiable tracking information. You may withdraw and cease participation in the study at any time without negative consequences. The final published results of the research will be aggregated measures and there will be no features that could identify individual participants.

The completion of the questionnaire indicates your consent to participate in the research entitled "Re-designing Help Desk's Support Process using Knowledge Management Framework", conducted by Nelson K. Y. Leung as it has been described to you in the information sheet and in discussion with Nelson K. Y. Leung. You understand that the data collected from your participation will be used for of master thesis, conference paper as well as journal paper publications, and you consent for it to be used in that manner.

If you have any enquiries about the research, you can contact Nelson K. Y. Leung on email: kn1164@uow.edu.au (Telephone number: 04-22217737) and Dr. Sim Kim Lau on email: simlau@uow.edu.au (Telephone number: 02-42214132). If you have any concerns or complaints regarding the way in which the research is or has been conducted, you should contact the Ethics Officer on (02) 4221 4457.

Thank you very much in anticipation of your willingness to participate in this study.

Questionnaire

1) How many individual users does your Help Desk support?

2) How many staff does your Help Desk employ?

Full time

Part time

3) What are your Help Desk operational hours?

Monday-Friday:

From _____ to _____

Saturday:

From _____ to _____

Sunday:

From _____ to _____

Public Holiday:

From _____ to _____

4) Which one(s) is/are the best to describe your Help Desk support model?

(CHECK ALL THAT APPLY)

- ☐ Decentralised Help Desk
- ☐ Single Point of Contact
- ☐ Distributed Help Desk
- ☐ Outsourcing
- ☐ e-support
- ☐ Other (please specify)

5) Which one is the best to describe your Help Desk support structure?

(CHECK ONLY ONE ANSWER)

- ☐ One Level Support
- ☐ Two Levels Support
- ☐ Three Levels Support
- ☐ Other (please specify)

6) Does your Help Desk currently use the following system or software?

(CHECK ALL THAT APPLY)

- ☐ Automatic Call Distributor (ACD) System
- ☐ Interactive Voice Response (IVR) System
- ☐ Help Desk Management System
- ☐ Expert System
- ☐ Remote Control System
- ☐ Knowledge Management System
- ☐ Internet / Web Interface
- ☐ Other (please specify)

7) Which of the following Administrative Issue(s) can be resolved by user if sufficient information is provided:

(CHECK ALL THAT APPLY)

- | | |
|--|--|
| <input type="checkbox"/> Account Setup | <input type="checkbox"/> Password Retrieval |
| <input type="checkbox"/> Account Termination | <input type="checkbox"/> Password Reset |
| <input type="checkbox"/> Account Maintenance | <input type="checkbox"/> Password Syntax information |
| <input type="checkbox"/> Account Login Problem | <input type="checkbox"/> Password Invalid |
| <input type="checkbox"/> Account Suspension | <input type="checkbox"/> Other (please specify) |

8) Which of the following Guideline(s) (in electronic format) should be provided to user if needed:

(CHECK ALL THAT APPLY)

- | | |
|--|---|
| <input type="checkbox"/> Hardware Installation | <input type="checkbox"/> Software Installation |
| <input type="checkbox"/> Software Purchasing | <input type="checkbox"/> Hardware Purchasing |
| <input type="checkbox"/> Service Purchasing | <input type="checkbox"/> Other (please specify) |

- 9) Assuming the user has sufficient guidelines, which of the following Hardware Problem(s) should the user attempt to solve before using the Help Desk:

(CHECK ALL THAT APPLY)

Hardware:

- ☐ CD / DVD ROM
- ☐ Printer
- ☐ Monitor
- ☐ Mouse
- ☐ Keyboard

- ☐ Scanner
- ☐ Hard Drive Tower
- ☐ Phone Headset
- ☐ Phone Handset
- ☐ Other (please specify)

- 10) Assuming the user has sufficient guidelines, which of the following Software Problem(s) should the user attempt to solve before using the Help Desk:

(CHECK ALL THAT APPLY)

Software:

- ☐ Software Performance
- ☐ Software “Can’t Start”

- ☐ Software Functionality
- ☐ Other (please specify)

- 11) Assuming the user has sufficient guidelines, which of the following “Other” Problem(s) should the user attempt to solve before using the Help Desk:

(CHECK ALL THAT APPLY)

Other:

- ☐ Website “Too Slow”
- ☐ Website “Unreachable”
- ☐ File “Missing”
- ☐ Other (please specify)

- ☐ Server “Too Slow”
- ☐ Server “Unreachable”
- ☐ File “Corruption”

- 12) ***Please specify that your information provided for Question 13 to Question 18 is based on:

(CHECK ONLY ONE ANSWER)

- ☐ Management Report

- ☐ Estimation

- 13) What is the total number of incoming telephone calls logged per month for the past three calendar months?

Month

Number of Calls

- 14) What is the average number of enquiries logged by the Help Desk Management System per month for the past three calendar months?

Month

Number of Calls

- 15) Over the past 12 months, has your Help Desk experienced ____ in the total amount of incoming enquiries?

(CHECK ONLY ONE ANSWER)

- ☐ An Increase
☐ A Decrease
☐ No Change

What are the three main reasons for the above selection?

- a) _____
b) _____
c) _____

- 16) Of your total user contact, what percentage is via:

- | | |
|---------------------------|---------|
| a) Telephone | _____ % |
| b) Walk-in | _____ % |
| c) Fax | _____ % |
| d) Internet/Email | _____ % |
| e) Other (please specify) | _____ % |

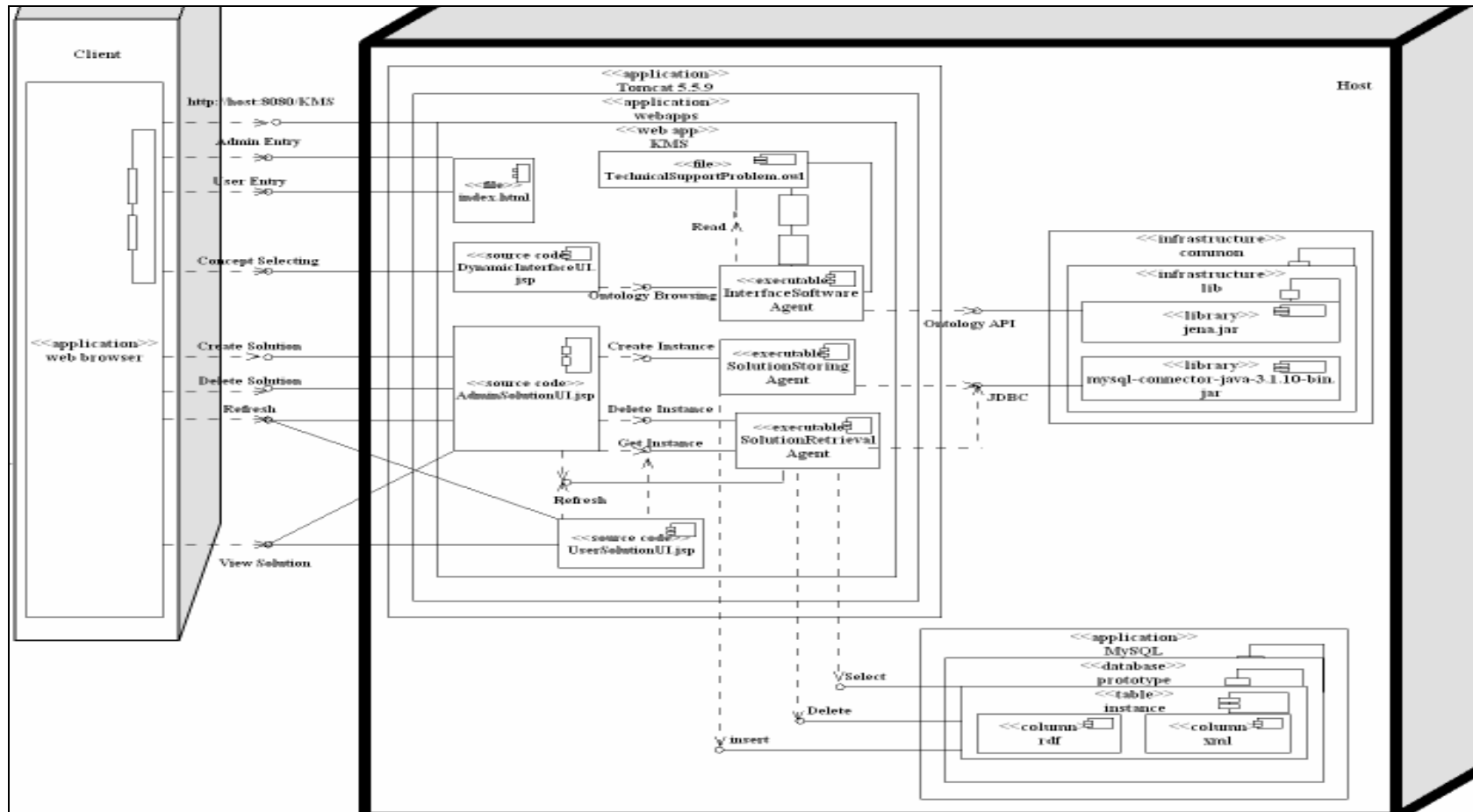
17) Of your total enquiries received by the Help Desk, what percentage is:

- | | |
|-------------------------------------|---------|
| a) Resolved by first level support | _____ % |
| b) Resolved by second level support | _____ % |
| c) Resolved by third level support | _____ % |
| d) Resolved by vendors | _____ % |
| e) Resolved by other | _____ % |

18) What mix of enquiries does your Help Desk get in each of the following area:

- | | |
|-------------------------------------|---------|
| a) Hardware / Software Installation | _____ % |
| b) Other Hardware Problem | _____ % |
| c) Other Software Problem | _____ % |
| d) Data Communications | _____ % |
| e) Voice Communications | _____ % |
| f) Account / Password | _____ % |
| g) Other (please specify) | _____ % |

Appendix B – Physical Design of the Prototype



Appendix C – Seventy Sets of Enquiry Types and their Symptoms

First Selection	Second Selection	Third Selection	Fourth Selection	Fifth Selection
Hardware_Problem	Non_Standard_Hardware_Problem	Vendor		
Hardware_Problem	Non_Standard_Hardware_Problem	Help Desk		
Hardware_Problem	Standard_Hardware_Problem	Mouse_Problem	Cursor_Frozen	
Hardware_Problem	Standard_Hardware_Problem	Mouse_Problem	Mouse_Button_Not_Responding	
Hardware_Problem	Standard_Hardware_Problem	Mouse_Problem	Mouse_Movement_Too_Slow	
Hardware_Problem	Standard_Hardware_Problem	Mouse_Problem	Mouse_Movement_Too_Fast	
Hardware_Problem	Standard_Hardware_Problem	Printer_Problem	No_Printout	
Hardware_Problem	Standard_Hardware_Problem	Printer_Problem	Abnormal_Printout	
Hardware_Problem	Standard_Hardware_Problem	Printer_Problem	Toner_Level_Low	
Hardware_Problem	Standard_Hardware_Problem	Monitor_Problem	Blackspot	
Hardware_Problem	Standard_Hardware_Problem	Monitor_Problem	Abnormal_Image	
Hardware_Problem	Standard_Hardware_Problem	Monitor_Problem	Screen_Flipping	
Hardware_Problem	Standard_Hardware_Problem	Monitor_Problem	No_Image	
Hardware_Problem	Standard_Hardware_Problem	Hard_Drive_Problem	Hard_Drive_Cannot_Start	
Hardware_Problem	Standard_Hardware_Problem	Hard_Drive_Problem	Hard_Drive_Cannot_Shut_Down	
Hardware_Problem	Standard_Hardware_Problem	Hard_Drive_Problem	Hard_Drive_Plugin_Not_Responding	
Hardware_Problem	Standard_Hardware_Problem	Hard_Drive_Problem	Hard_Drive_Hung	
Hardware_Problem	Standard_Hardware_Problem	Hard_Drive_Problem	Hard_Drive_Overheat	
Hardware_Problem	Standard_Hardware_Problem	Keyboard_Problem	Particular_Key_Not_Responding	
Hardware_Problem	Standard_Hardware_Problem	Keyboard_Problem	Keyboard_Long_Delay	
Hardware_Problem	Standard_Hardware_Problem	Keyboard_Problem	Entire_Keyboard_Not_Responding	
Hardware_Problem	Standard_Hardware_Problem	CD_DVD_ROM_Problem	ROM_Cannot_Record	
Hardware_Problem	Standard_Hardware_Problem	CD_DVD_ROM_Problem	ROM_Cannot_Play	
Hardware_Problem	Standard_Hardware_Problem	CD_DVD_ROM_Problem	ROM_Cannot_Close	
Hardware_Problem	Standard_Hardware_Problem	CD_DVD_ROM_Problem	ROM_Cannot_Open	
Software_Problem	Performance_Problem	Software_Cannot_Start		
Software_Problem	Performance_Problem	Software_Frozen		
Software_Problem	Performance_Problem	Software_Slow_Performance		
Software_Problem	Performance_Problem	Software_Abnormal_Performance		

Software_Problem	Functional_Problem	Standard_Software_Problem	Internet_Explorer_Problem	
Software_Problem	Functional_Problem	Standard_Software_Problem	McAfee_Virus_Scan_Problem	
Software_Problem	Functional_Problem	Standard_Software_Problem	MS_Office_Problem	MS_Outlook_Problem
Software_Problem	Functional_Problem	Standard_Software_Problem	MS_Office_Problem	MS_Access_Problem
Software_Problem	Functional_Problem	Standard_Software_Problem	MS_Office_Problem	MS_PowerPoint_Problem
Software_Problem	Functional_Problem	Standard_Software_Problem	MS_Office_Problem	MS_Word_Problem
Software_Problem	Functional_Problem	Standard_Software_Problem	MS_Office_Problem	MS_Excel_Problem
Software_Problem	Functional_Problem	Standard_Software_Problem	Adobe_PDF_Problem	
Software_Problem	Functional_Problem	Non_Standard_Software_Problem	Vendor	
Software_Problem	Functional_Problem	Non_Standard_Software_Problem	Help Desk	
IT_Administrative_Issue	Hardware_Installation_Guidance	Non_Standard_Hardware_Installation_Guidance		
IT_Administrative_Issue	Hardware_Installation_Guidance	Standard_Hardware_Installation_Guidance		
IT_Administrative_Issue	Account_Issue	PC_Account_Issue	PC_Account_Termination	
IT_Administrative_Issue	Account_Issue	PC_Account_Issue	PC_Account_Setup	
IT_Administrative_Issue	Account_Issue	PC_Account_Issue	PC_Account_Cannot_Login	
IT_Administrative_Issue	Account_Issue	PC_Account_Issue	PC_Account_Suspension	
IT_Administrative_Issue	Account_Issue	Email_Account_Issue	Email_Account_Suspension	
IT_Administrative_Issue	Account_Issue	Email_Account_Issue	Email_Account_Setup	
IT_Administrative_Issue	Account_Issue	Email_Account_Issue	Email_Account_Maintenance	
IT_Administrative_Issue	Account_Issue	Email_Account_Issue	Email_Account_Termination	
IT_Administrative_Issue	Account_Issue	Email_Account_Issue	Email_Account_Cannot_Login	
IT_Administrative_Issue	Password_Issue	Retrieve_Password		
IT_Administrative_Issue	Password_Issue	Reset_Password		
IT_Administrative_Issue	Password_Issue	Password_Syntax_Info		
IT_Administrative_Issue	Password_Issue	Invalid_Password		
IT_Administrative_Issue	Password_Issue	Change_Password		
IT_Administrative_Issue	IT_Product_Purchasing_Guidance			
IT_Administrative_Issue	Equipment_Moving_Guidance			
IT_Administrative_Issue	Software_Installation_Guidance	Non_Standard_Software_Installation_Guidance		
IT_Administrative_Issue	Software_Installation_Guidance	Standard_Software_Installation_Guidance		
Other_Problem	Website_Problem	Enterprise_Website_Problem	Abnormal_Website>Loading_Speed	
Other_Problem	Website_Problem	Enterprise_Website_Problem	Website_Cannot_Completely_Load	
Other_Problem	Website_Problem	Enterprise_Website_Problem	Website_Error_Message	
Other_Problem	Website_Problem	Non_Enterprise_Website_Problem		
Other_Problem	File_Problem	File_Corrupted		

Other_Problem	File_Problem	File_Accidentally_Deleted		
Other_Problem	File_Problem	File_Accidentally_Modified		
Other_Problem	File_Problem	Missing_File		
Other_Problem	Remote_Server_Problem	Server_Slow		
Other_Problem	Remote_Server_Problem	Missing_Folder		
Other_Problem	Remote_Server_Problem	Cannot_Login_to_Server		

Appendix D – Program Codes

The order of files are in alphabetical ascending.

AdminSolutionUI.jsp
DynamicInterfaceUI.jsp
Index.html
Instance.java
InterfaceSoftwareAgent.java
SolutionRetrievalAgent.java
SolutionStoringAgent.java
TechnicalSupportProblem.owl
UserSolutionUI.jsp
web.xml

AdminSolutionUI.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.util.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Ontology browsing</title>
</head>
<body>
Help Desk Inquiry
<%
HashMap list = (HashMap) session.getAttribute("list");
if (list!=null){
Iterator iter = list.keySet().iterator();
Iterator itNext = list.keySet().iterator();
itNext.next();
Object nextKey = new Object();
%>
<form method="get" action="/KMS/SolutionRetrievalAgent">
<%
while (iter.hasNext()){
    Object keyTitle = iter.next();
    Hashtable ht = (Hashtable) list.get(keyTitle);
    if (ht!=null){
%>
<br/>
<%if (ht.keys().hasMoreElements()){ %>
<select
<% if (!iter.hasNext()) {%>
onChange="location=this.options[this.selectedIndex].value;">
<%} else {%>><%} %>
<option></option>
<%
        Enumeration it = ht.keys();
        if (itNext.hasNext()) nextKey = itNext.next();
        if (it!=null) while (it.hasMoreElements()){
            Object key = it.nextElement();
            System.out.println(key+" "+nextKey );
%>
<option
<%if (nextKey.toString().equals(key.toString())) {%>
selected="selected"
<%} %>
value="/KMS/InterfaceSoftwareAgent?user=admin&useraction=welcome&
concept=<%=ht.get(key)%>&ontology=<%=key %>"><%=ht.get(key)%></option>
<%} %>
</select>
<br/>
```

```

<%}}} %>
<%
String solution = request.getAttribute("solution").toString();
String problem = request.getAttribute("problem").toString();
if ((!solution.equals(""))&&(!problem.equals(""))){
%>
Solution:<br/><pre>
<%=request.getAttribute("solution").toString() %>
</pre>
<br/><br/><br/>
Problem:<br/><pre>
<%=request.getAttribute("problem").toString() %>
</pre>
<br/>
<input type="submit" name="useraction" value="Delete"/>
<input type="submit" name="useraction" value="Refresh"/>
</form>
<%=} else{ %>
<input type="submit" name="useraction" value="Refresh"/>
</form>
<form method="get" action="/KMS/SolutionStoringAgent">
Problem:
<br/>
<textarea name="problem" cols="30" rows="4"></textarea>
<br/>
Solution:<br/>
<textarea name="solution" cols="30" rows="4"></textarea>
<br/>
<input type="submit" name="useraction" value="OK"/>
</form>
<%=}}} %>
</body>
</html>

```

DynamicInterfaceUI.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.util.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Ontology browsing</title>
</head>
<body>
Help Desk Inquiry
<%
HashMap list = (HashMap) session.getAttribute("list");
if (list!=null){
Iterator iter = list.keySet().iterator();
Iterator itNext = list.keySet().iterator();
itNext.next();
Object nextKey = new Object();
%>
<from method="get">
<%
while (iter.hasNext()){
    Object keyTitle = iter.next();
    Hashtable ht = (Hashtable) list.get(keyTitle);
    if (ht!=null){
%>

<br/>
<select
<% if (!iter.hasNext()) {%>
onChange="location=this.options[this.selectedIndex].value;">
<%} else {%>><%} %>
<option></option>
<%
        Enumeration it = ht.keys();
        if (itNext.hasNext()) nextKey = itNext.next();
        if (it!=null) while (it.hasMoreElements()){
            Object key = it.nextElement();
            System.out.println(key+" "+nextKey );
%>
<option
<%if (nextKey.toString().equals(key.toString())) {%>
selected="selected"
<%} %>
value="/KMS/InterfaceSoftwareAgent?useraction=welcome&
concept=<%=ht.get(key)%>&ontology=<%=key %>"><%=ht.get(key)%></option>
<%} %>
</select>
<br/>
```

```
<%}} %>
</form>
<%} %>
</body>
</html>
```

Index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Knowledge Management System</title>
</head>
<body>
<br/>
<a href="/KMS/InterfaceSoftwareAgent?user=admin&useraction=login">Admin
Entry</a>
<br/>
<a href="/KMS/InterfaceSoftwareAgent?user=user&useraction=login">User
Entry</a>
</body>
</html>
```

Instance.java

```
import java.io.Serializable;

/**
 * Utility class that helps generate RDF and XML, retrieve values from RDF and
 * XML.
 * @author
 *
 */
public class Instance implements Serializable{

    /**
     * Needed for persistance storage of session
     */
    private static final long serialVersionUID = -3913236659784035419L;
    /**
     * Regular XML header. Used for XML generation
     */
    private static final String xmlHeader =
        "<?xml version='1.0'?><xml>";
    /**
     * Regular XML footer. USed for XML generation
     */
    private static final String xmlFooter =
        "</xml>";
    /**
     * RDF header for instance description with import of needed ontologies,
     * standards and schemas.
     */
    private static final String rdfHeader =
        "<?xml version='1.0'?>" +
        "<rdf:RDF" +
        "  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#" +
        "  xmlns:xsd='http://www.w3.org/2001/XMLSchema#" +
        "  xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#" +
        "  xmlns:owl='http://www.w3.org/2002/07/owl#" +
        "  xmlns='http://www.owl-ontologies.com/unnamed.owl#" +
        "  xml:base='http://www.owl-ontologies.com/unnamed.owl'>";
    /**
     * Regular RDF footer.
     */
    private static final String rdfFooter = "</rdf:RDF>";
    /**
     * A direct class of an instance.
     */
    public String classOWL = "";
    /**
     * An object property of an instance.
     */
}
```

```

    */
    public String propertyOWL = "";
    /**
     * A value of a property of an instance.
     */
    public String propertyValue = "";
    /**
     * Description of problem for an instance.
     */
    public String problem = "";
    /**
     * Description of a solution for an instance.
     */
    public String solution = "";

    /**
     * Generates XML document with description of problem and solution.
     * @return XML document
     */
    public String getXML() {
        String result = "";
        result+=xmlHeader;
        result+="<problem>" + problem + "</problem>";
        result+="<solution>" + solution + "</solution>";
        result+=xmlFooter;
        return result;
    }
    /**
     * Generates RDF document with class, ID, property and property's value.
     * @return RDF document
     */
    public String getRDF() {
        String result = "";
        result+=rdfHeader;
        result = result + "<" + classOWL + " rdf:ID=\"" + classOWL;
        if (!propertyOWL.equals("")) {
            result= result + propertyOWL + propertyValue + ">";
            result= result + "<" + propertyOWL + "
rdf:resource='#" + propertyValue + "'/>";
        } else result = result + ">";
        result = result + "</" + classOWL + ">";
        result+=rdfFooter;
        return result;
    }
    /**
     * Parses XML document to assign problem and solution fields.
     * @param xml document to parse.
     */
    public void setXML(String xml) {
        int solutionIndexBegin = xml.indexOf("<solution>")+10;

```



```

        int solutionIndexEnd = xml.indexOf("</solution>");
        solution = xml.substring(solutionIndexBegin, solutionIndexEnd);

        int problemIndexBegin = xml.indexOf("<problem>")+9;
        int problemIndexEnd = xml.indexOf("</problem>");
        problem = xml.substring(problemIndexBegin, problemIndexEnd);
    }
    /**
     * Parses RDF document to assign class, property and property value fields.
     * @param rdf document to parse.
     */
    public void setRDF(String rdf) {
        int classIndexBegin = rdf.indexOf("<",30)+1;
        int classIndexEnd = rdf.indexOf(" ", classIndexBegin);
        classOWL = rdf.substring(classIndexBegin, classIndexEnd);

        int propertyIndexBegin = rdf.indexOf("<", classIndexBegin)+1;
        int propertyIndexEnd = rdf.indexOf(" ", propertyIndexBegin);
        propertyOWL = rdf.substring(propertyIndexBegin, propertyIndexEnd);

        int propertyValueIndexBegin = rdf.indexOf("#",
propertyIndexBegin)+1;
        int propertyValueIndexEnd = rdf.indexOf(">",
propertyValueIndexBegin)-1;
        propertyValue = rdf.substring(propertyValueIndexBegin,
propertyValueIndexEnd);
    }

```

InterfaceSoftwareAgent.java

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedHashMap;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.ontology.OntProperty;
import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.util.iterator.ExtendedIterator;

/**
 * Servlet implementation class for InterfaceSoftwareAgent
 *
 * @web.servlet
 *   name="InterfaceSoftwareAgent"
 *   display-name="InterfaceSoftwareAgent"
 *
 * @web.servlet-mapping
 *   url-pattern="/InterfaceSoftwareAgent"
 */
public class InterfaceSoftwareAgent extends javax.servlet.http.HttpServlet
implements javax.servlet.Servlet {
    /**
     *
     */
    /**
     * private static final long serialVersionUID = 3604763032423410468L;
     */
    /**
     * Jena model of the ontology
     */
    private OntModel ontModel;
    /**
     * root class that equals http://www.owl-
    ontologies.com/unnamed.owl#Help_Desk_Enquiry in this ontology
     */
    private OntClass root;
```

```

    /* (non-Java-doc)
    * @see javax.servlet.http.HttpServlet#HttpServlet()
    */
    public InterfaceSoftwareAgent() {
        super();
    }

    /* (non-Java-doc)
    * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
    HttpServletResponse response)
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        processRequest(request, response);
    }

    /* (non-Java-doc)
    * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
    HttpServletResponse response)
    */
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        processRequest(request, response);
    }

}
/**
 * Processes user requests.
 * @param request HTTP request from GET and POST methods
 * @param response HTTP response
 * @throws ServletException
 * @throws IOException
 */
private void processRequest(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String buildURL = "";
    System.out.println("Entering processRequest of
InterfaceSoftwareAgent");
    HttpSession session = request.getSession();
    String useraction = request.getParameter("useraction");
    String user = request.getParameter("user");
    OntResource currentConceptResource;
    OntClass currentConceptClass;
    Instance instance = initInstance(session);
    LinkedHashMap list = initList(session);
    String concept = initConcept(request);
    if (useraction.equals("login")) {
        session.setAttribute("user", user);
        list = new LinkedHashMap();
        useraction = "welcome";
    }
}

```

```

//Building subclasses
if (useraction.equals("welcome")){
    currentConceptResource =
initCurrentConceptResource(concept);
    updateList(currentConceptResource, list);
    Hashtable ht = new Hashtable();
    if (currentConceptResource.isClass()) {
        currentConceptClass =
currentConceptResource.asClass();
        ExtendedIterator it =
currentConceptClass.listSubClasses(true);
        if (it.hasNext()){//has subclasses
            addSubclasses(currentConceptClass, ht, it);
            buildURL="/DynamicInterfaceUI.jsp";
        }else{//no subclasses

            instance.classOWL=currentConceptClass.getLocalName();
            System.out.println("No subclasses");
            ExtendedIterator itInstances =
currentConceptClass.listInstances();
            if (itInstances.hasNext()){
                addInstances(currentConceptClass, ht,
itInstances);
                buildURL="/DynamicInterfaceUI.jsp";
            }else{//no instances
                System.out.println("No instances");
                ExtendedIterator itProperties =
currentConceptClass.listDeclaredProperties(false);
                if (itProperties.hasNext()){//has
properties
                    while (itProperties.hasNext()){
                        OntProperty property =
(OntProperty) itProperties.next();

                        System.out.println(property);

                        ExtendedIterator itRanges
= property.listRange();
                        if (itRanges.hasNext()){
                            OntClass
rangeClass = (OntClass) itRanges.next();

                            System.out.println("range "+ rangeClass);

                            ExtendedIterator
itRangeInstances = rangeClass.listInstances();
                            if
((!rangeClass.listSubClasses().hasNext())&(itRangeInstances.hasNext())){
                                instance.propertyOWL = property.getLocalName();

                                addInstances(rangeClass, ht, itRangeInstances);

```

```

        buildURL="/DynamicInterfaceUI.jsp";
                                                break;
    } else { //range class
does not have instances or has subclasses
                                                //forward
to SolutinRetrievalAgent

        buildURL="/SolutionRetrievalAgent";
                                                }
                                                }
    } else { //does not have properties
                                                //forward to
SolutinRetrievalAgent

        buildURL="/SolutionRetrievalAgent";
    }
    }
    } else { //currentConcept is not a class
        //forward to SolutinRetrievalAgent
        instance.propertyValue =
currentConceptResource.getLocalName();
        buildURL="/SolutionRetrievalAgent";
    }
    //forwarding
    list.put(currentConceptResource,ht);
    System.out.println(" --- ");
    session.setAttribute("list",list);
    session.setAttribute("instance", instance);
    RequestDispatcher rd =
getServletContext().getRequestDispatcher(buildURL);
    rd.forward(request, response);
}
}
/**
 * Initialized instance to maintain selections and to create, delete and search
XML in Database.
 * @param session HTTP session.
 * @return initialized instance.
 */
private Instance initInstance(HttpSession session) {
    Instance instance = (Instance) session.getAttribute("instance");
    if (instance==null) instance = new Instance();
    System.out.println("instance is " + instance.classOWL + "=" +
instance.propertyOWL);
    return instance;
}
/**

```

```

        * Updates list of selections. Needed to handle situation when user used
        "Back" button and selected
        * another concept.
        * @param currentConceptResource is a selected concept.
        * @param list of selected concepts.
        */
        private void updateList(OntResource currentConceptResource,
        LinkedHashMap list) {
            boolean flag = false;
            ArrayList keys = new ArrayList();
            Iterator iterator = list.keySet().iterator();
            while (iterator.hasNext()){
                Object testKey = iterator.next();
                Hashtable testValue = (Hashtable) list.get(testKey);
                if (flag) keys.add(testKey);
                if (testValue.containsKey(currentConceptResource)) flag =
true;
            }
            Iterator iterator2 = keys.iterator();
            while (iterator2.hasNext()){
                list.remove(iterator2.next());
            }
        }
        /**
        * Adds instances of property values for further selection by user.
        * @param currentConceptClass is a direct class for instances.
        * @param ht is a table to store full ID and human readable name.
        * @param itInstances is a iterator over instances.
        */
        private void addInstances(OntClass currentConceptClass, Hashtable ht,
        ExtendedIterator itInstances) {
            while (itInstances.hasNext()){
                Individual individual = (Individual) itInstances.next();
                System.out.println(individual+" == "+currentConceptClass);
                ht.put(individual, individual.getLocalName());
            }
        }
        /**
        * Adds subclasses of selected concept for further selection by user.
        * @param currentConceptClass is a direct superclass for subclasses.
        * @param ht is a table to store full ID and human readable name.
        * @param it s a iterator over subclasses.
        */
        private void addSubclasses(OntClass currentConceptClass, Hashtable ht,
        ExtendedIterator it) {
            while (it.hasNext()){
                OntClass subclass = (OntClass) it.next();
                System.out.println(subclass+" == "+currentConceptClass);
                if (!subclass.isAnon())
                    ht.put(subclass, subclass.getLocalName());
            }
        }

```

```

    }
}
/**
 * Initializes current concept.
 * @param concept is a selected concept.
 * @return current concept as OntResource
 */
private OntResource initCurrentConceptResource(String concept) {
    OntResource currentConcept;
    if(concept==null){
        currentConcept = root;
    }else{
        currentConcept = ontModel.getOntResource(concept);
    }
    System.out.println("currentConcept is " + currentConcept);
    return currentConcept;
}
/**
 * Initializes list of selected concepts.
 * @param session HTTP session.
 * @return initialized list.
 */
private LinkedHashMap initList(HttpSession session) {
    LinkedHashMap list = new LinkedHashMap();
    Object listTemp = session.getAttribute("list");
    if (listTemp!=null) list= (LinkedHashMap) listTemp;
    return list;
}
/**
 * Formats correct name of selected concept from HTTP request parameters.
 * @param request HTTP request.
 * @return correct name of selected concept.
 */
private String initConcept(HttpServletRequest request) {
    String concept = request.getParameter("concept");
    System.out.println("concept is " + concept);
    if (concept!=null) concept = request.getParameter("ontology") + "#" +
concept;
    System.out.println("full name of the concept is " + concept);
    return concept;
}

/* (non-Javadoc)
 * Initializes Jena model for ontology and root class.
 * @see javax.servlet.GenericServlet#init()
 */
public void init() throws ServletException {
    super.init();
    ontModel =
ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);

```

```
        ontModel.read("http://localhost:8080/KMS/TechnicalSupportProblem.owl");
        root = ontModel.getOntClass("http://www.owl-
ontologies.com/unnamed.owl#Help_Desk_Enquiry");
    }
}
```


SolutionRetrievalAgent.java

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
 * Servlet implementation class for SolutionRetrievalAgent
 *
 * @web.servlet
 *   name="SolutionRetrievalAgent"
 *   display-name="SolutionRetrievalAgent"
 *
 * @web.servlet-mapping
 *   url-pattern="/SolutionRetrievalAgent"
 *
 */
public class SolutionRetrievalAgent extends javax.servlet.http.HttpServlet
implements javax.servlet.Servlet {
    /**
     *
     */
    private static final long serialVersionUID = -2801682834749239489L;
    /**
     * Connection URL for mySQL database prototype that resides on localhost.
     */
    private String url = "jdbc:mysql://localhost/prototype";
    /**
     * Connection user name.
     */
    private String user = "root";
    /**
     * Connection password.
     */
    private String pass = "prototype";
    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#HttpServlet()
     */
    public SolutionRetrievalAgent() {
        super();
    }
}
```

```

        /* (non-Java-doc)
        * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
        HttpServletResponse response)
        */
        protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
            processRequest(request, response);
        }

        /* (non-Java-doc)
        * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
        HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
            processRequest(request, response);
        }
    }
    /**
    * Processes user requests.
    * @param request HTTP request from GET and POST methods
    * @param response HTTP response
    * @throws ServletException
    * @throws IOException
    */
    private void processRequest(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        String buildURL = "";
        System.out.println("Entering processRequest of
        SolutionRetrivalAgent");
        HttpSession session = request.getSession();
        String useraction = request.getParameter("useraction");
        String user = (String) session.getAttribute("user");
        Instance instance= (Instance) session.getAttribute("instance");
        if (user.equals("admin")){
            if (useraction.equals("welcome")){
                String xml = getInstance(instance);
                if (xml!=null){
                    instance.setXML(xml);
                    request.setAttribute("solution", instance.solution);
                    request.setAttribute("problem", instance.problem);
                    buildURL="/AdminSolutionUI.jsp";
                } else {
                    request.setAttribute("solution", "");
                    request.setAttribute("problem", "");
                    buildURL="/SolutionStoringAgent";
                }
            }
        }
        if (useraction.equals("Delete")){
            deleteInstance(instance);
        }
    }

```

```

        request.setAttribute("solution", "");
        request.setAttribute("problem", "");
        buildURL="/SolutionStoringAgent";
    }
    if (useraction.equals("Refresh")){
        session.invalidate();
        buildURL =
"/InterfaceSoftwareAgent?user=admin&useraction=login";
    }
    }else{
        if (useraction.equals("welcome")){
            String xml = getInstance(instance);
            if (xml!=null){
                instance.setXML(xml);
                request.setAttribute("solution",
instance.solution);
                request.setAttribute("problem",
instance.problem);
                buildURL="/UserSolutionUI.jsp";
            }else{
                request.setAttribute("solution", "");
                request.setAttribute("problem", "");
                buildURL="/UserSolutionUI.jsp";
            }
        }
        if (useraction.equals("Refresh")){
            session.invalidate();
            buildURL =
"/InterfaceSoftwareAgent?user=user&useraction=login";
        }
    }
    RequestDispatcher rd =
getServletContext().getRequestDispatcher(buildURL);
    rd.forward(request, response);
}
/**
 * Deletes entry in database for specified instance.
 * @param instance to delete.
 */
private void deleteInstance(Instance instance) {
    PreparedStatement stmt = null;

    try {
        Connection conn = DriverManager.getConnection(url, user,
pass);
        stmt = conn.prepareStatement("DELETE FROM
instance WHERE rdf=?");
        stmt.setString(1, instance.getRDF());

```

```

        System.out.println("deleting DB for
"+instance.getRDF());
        stmt.execute();

        stmt.close();
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
/**
 * Retrieves instance from database.
 * @param instance to retrieve.
 * @return XML with description of problem and solution.
 */
private String getInstance(Instance instance) {
    PreparedStatement stmt = null;
    ResultSet rs = null;
    String xml = null;
    try {
        Connection conn = DriverManager.getConnection(url, user,
pass);
        stmt = conn.prepareStatement("SELECT xml FROM instance
WHERE rdf=?");
        stmt.setString(1,instance.getRDF());

        System.out.println("looking DB for "+instance.getRDF());

        if (stmt.execute()){
            rs = stmt.getResultSet();
        }
        if (rs.next()){
            xml = rs.getString("xml");
        }
        rs.close();
        stmt.close();
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return xml;
}

/* (non-Javadoc)
 * @see javax.servlet.GenericServlet#init()
 */
public void init() throws ServletException {

```

```

super.init();
try {
    Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (InstantiationException e) {
    e.printStackTrace();
} catch (IllegalAccessException e) {
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
}
}

```

SolutionStoringAgent.java

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class for SolutionStoringAgent
 *
 * @web.servlet
 *   name="SolutionStoringAgent"
 *   display-name="SolutionStoringAgent"
 *
 * @web.servlet-mapping
 *   url-pattern="/SolutionStoringAgent"
 *
 */
public class SolutionStoringAgent extends javax.servlet.http.HttpServlet implements
javax.servlet.Servlet {
    /**
     *
     */
    private static final long serialVersionUID = -5279532178834123159L;
    /**
     * Connection URL for mySQL database prototype that resides on localhost.
     */
    private String url = "jdbc:mysql://localhost/prototype";
    /**
     * Connection user name.
     */
    private String user = "root";
    /**
     * Connection password.
     */
    private String pass = "prototype";
    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#HttpServlet()
     */
    public SolutionStoringAgent() {
        super();
    }
}
```

```

        /* (non-Java-doc)
        * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
        HttpServletResponse response)
        */
        protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
            processRequest(request, response);
        }

        /* (non-Java-doc)
        * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
        HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
            processRequest(request, response);
        }
    }
    /**
    * Processes user requests.
    * @param request HTTP request from GET and POST methods
    * @param response HTTP response
    * @throws ServletException
    * @throws IOException
    */
    private void processRequest(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        String buildURL = "/AdminSolutionUI.jsp";
        HttpSession session = request.getSession();
        String useraction = request.getParameter("useraction");
        String solution = request.getParameter("solution");
        if (solution==null) solution = "";
        String problem = request.getParameter("problem");
        if (problem==null) problem = "";
        Instance instance= (Instance) session.getAttribute("instance");
        if (useraction.equals("OK")){
            instance.problem = problem;
            instance.solution = solution;
            if (!solution.equals("")&&!problem.equals(""))
                createInstance(instance);
            session.invalidate();

            buildURL="/InterfaceSoftwareAgent?user=admin&useraction=login";
        }
        RequestDispatcher rd =
        getServletContext().getRequestDispatcher(buildURL);
        rd.forward(request, response);
    }
    /**
    * Creates new entry with specified instance in Database.

```

```

    * @param instance to create.
    */
    private void createInstance(Instance instance) {
        PreparedStatement stmt = null;

        try {
            Connection conn = DriverManager.getConnection(url, user,
pass);
                stmt = conn.prepareStatement("INSERT INTO
instance(rdf, xml) VALUES (?,?)");
                stmt.setString(1,instance.getRDF());
                stmt.setString(2,instance.getXML());
                stmt.execute();
                //process result
                stmt.close();
                conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

    }

    /* (non-Javadoc)
    * @see javax.servlet.GenericServlet#init()
    */
    public void init() throws ServletException {
        super.init();
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}

```


TechnicalSupportProblem.owl

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="PC_Account_Issue_Symptom">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Account_Issue_Symptom"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Website_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Other_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Hardware_Problem_Symptom">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Problem_Symptom"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="IT_Administration_Issue">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Help_Desk_Enquiry"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Printer_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Standard_Hardware_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MS_Office_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Standard_Software_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Standard_Hardware_Installation_Guidance">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Hardware_Installation_Guidance"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Software_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Problem_Symptom"/>
  </owl:Class>
  <owl:Class rdf:ID="Software_Problem">
```

```

    <rdfs:subClassOf rdf:resource="#Help_Desk_Enquiry"/>
  </owl:Class>
  <owl:Class rdf:ID="Non-Enterprise_Website_Problem">
    <rdfs:subClassOf rdf:resource="#Website_Problem"/>
  </owl:Class>
  <owl:Class rdf:ID="Performance_Problem">
    <rdfs:subClassOf rdf:resource="#Software_Problem"/>
  </owl:Class>
  <owl:Class rdf:ID="Functional_Problem">
    <rdfs:subClassOf rdf:resource="#Software_Problem"/>
  </owl:Class>
  <owl:Class rdf:ID="Remote_Server_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Other_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Password_Issue_Symptom">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="IT_Administration_Issue_Symptom"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Non-Standard_Software_Installation_Guidance">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Software_Installation_Guidance"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Monitor_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Standard_Hardware_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Website_Problem_Symptom">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Other_Problem_Symptom"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#IT_Administration_Issue_Symptom">
    <rdfs:subClassOf rdf:resource="#Problem_Symptom"/>
  </owl:Class>
  <owl:Class rdf:about="#Standard_Hardware_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Hardware_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Other_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Problem_Symptom"/>
  </owl:Class>
  <owl:Class rdf:ID="File_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Other_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>

```

```

    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Email_Account_Issue_Symptom">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Account_Issue_Symptom"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Keyboard_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
  </owl:Class>
  <owl:Class rdf:ID="CD_DVD_ROM_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
  </owl:Class>
  <owl:Class rdf:ID="Remote_Server_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Other_Problem_Symptom"/>
  </owl:Class>
  <owl:Class rdf:ID="Keyboard_Problem_Symptom">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Standard_Hardware_Problem_Symptom"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="IT_Product_Purchasing_Guidance">
    <rdfs:subClassOf rdf:resource="#IT_Administration_Issue"/>
  </owl:Class>
  <owl:Class rdf:ID="PC_Account_Issue">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Account_Issue"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Standard_Hardware_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Hardware_Problem_Symptom"/>
  </owl:Class>
  <owl:Class rdf:ID="Standard_Software_Installation_Guidance">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Software_Installation_Guidance"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Password_Issue">
    <rdfs:subClassOf rdf:resource="#IT_Administration_Issue"/>
  </owl:Class>
  <owl:Class rdf:ID="Equipment_Moving_Guidance">
    <rdfs:subClassOf rdf:resource="#IT_Administration_Issue"/>
  </owl:Class>
  <owl:Class rdf:ID="Internet_Explorer_Problem">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Standard_Software_Problem"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Mouse_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>

```

```

</owl:Class>
<owl:Class rdf:ID="Monitor_Problem_Symptom">
  <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:about="#Account_Issue_Symptom">
  <rdfs:subClassOf rdf:resource="#IT_Administration_Issue_Symptom"/>
</owl:Class>
<owl:Class rdf:about="#Hardware_Installation_Guidance">
  <rdfs:subClassOf rdf:resource="#IT_Administration_Issue"/>
</owl:Class>
<owl:Class rdf:ID="Non-Standard_Hardware_Installation_Guidance">
  <rdfs:subClassOf rdf:resource="#Hardware_Installation_Guidance"/>
</owl:Class>
<owl:Class rdf:about="#Account_Issue">
  <rdfs:subClassOf rdf:resource="#IT_Administration_Issue"/>
</owl:Class>
<owl:Class rdf:ID="Non-Standard_Hardware_Problem">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Non_Standard_Problem"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Hardware_Problem"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Standard_Software_Problem">
  <rdfs:subClassOf rdf:resource="#Functional_Problem"/>
</owl:Class>
<owl:Class rdf:ID="Enterprise_Website_Symptom">
  <rdfs:subClassOf rdf:resource="#Website_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="Installer"/>
<owl:Class rdf:about="#Software_Installation_Guidance">
  <rdfs:subClassOf rdf:resource="#IT_Administration_Issue"/>
</owl:Class>
<owl:Class rdf:ID="Hard_Drive_Problem_Symptom">
  <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="File_Problem_Symptom">
  <rdfs:subClassOf rdf:resource="#Other_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="Hard_Drive_Problem">
  <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
</owl:Class>
<owl:Class rdf:ID="Enterprise-Website_Problem">
  <rdfs:subClassOf rdf:resource="#Website_Problem"/>
</owl:Class>
<owl:Class rdf:about="#Other_Problem">
  <rdfs:subClassOf rdf:resource="#Help_Desk_Enquiry"/>
</owl:Class>
<owl:Class rdf:ID="Printer_Problem_Symptom">

```

```

    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="McAfee_Virus_Scan_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Software_Problem"/>
</owl:Class>
<owl:Class rdf:about="#Hardware_Problem">
    <rdfs:subClassOf rdf:resource="#Help_Desk_Enquiry"/>
</owl:Class>
<owl:Class rdf:ID="Software_Performance_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Software_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="Email_Account_Issue">
    <rdfs:subClassOf rdf:resource="#Account_Issue"/>
</owl:Class>
<owl:Class rdf:ID="Non-Standard_Software_Problem">
    <rdfs:subClassOf rdf:resource="#Non_Standard_Problem"/>
    <rdfs:subClassOf rdf:resource="#Functional_Problem"/>
</owl:Class>
<owl:Class rdf:ID="CD_DVD_ROM_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="Mouse_Problem_Symptom">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem_Symptom"/>
</owl:Class>
<owl:Class rdf:ID="Adobe_PDF_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Software_Problem"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasHard_Drive_Symptom">
    <rdfs:range rdf:resource="#Hard_Drive_Problem_Symptom"/>
    <owl:inverseOf>
        <owl:ObjectProperty rdf:ID="inverse_of_hasHard_Drive_Symptom"/>
    </owl:inverseOf>
    <rdfs:subPropertyOf>
        <owl:ObjectProperty rdf:ID="hasHardwareSymptom"/>
    </rdfs:subPropertyOf>
    <rdfs:domain rdf:resource="#Hard_Drive_Problem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPrinterSymptom">
    <rdfs:subPropertyOf>
        <owl:ObjectProperty rdf:about="#hasHardwareSymptom"/>
    </rdfs:subPropertyOf>
    <rdfs:domain rdf:resource="#Printer_Problem"/>
    <rdfs:range rdf:resource="#Printer_Problem_Symptom"/>
    <owl:inverseOf>
        <owl:ObjectProperty rdf:ID="inverse_of_hasPrinterSymptom"/>
    </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasSoftwareSymptom">
    <rdfs:domain rdf:resource="#Software_Problem_Symptom"/>
    <rdfs:range rdf:resource="#Software_Problem"/>

```

```

<owl:inverseOf>
  <owl:ObjectProperty rdf:ID="hasSoftwareSymptom"/>
</owl:inverseOf>
<rdfs:subPropertyOf>
  <owl:ObjectProperty rdf:ID="isSymptomOf"/>
</rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasKeyboardSymptom">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasHardwareSymptom"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Keyboard_Problem"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="inverse_of_hasKeyboardSymptom"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Keyboard_Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasPerformanceSymptom">
  <rdfs:subPropertyOf rdf:resource="#inverse_of_hasSoftwareSymptom"/>
  <rdfs:range rdf:resource="#Performance_Problem"/>
  <rdfs:domain rdf:resource="#Software_Performance_Problem_Symptom"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasPerformanceSymptom"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRemoteServerSymptom">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="hasOtherProblemSymptom"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Remote_Server_Problem"/>
  <rdfs:range rdf:resource="#Remote_Server_Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasFileSymptom">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasFileSymptom"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#File_Problem_Symptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="inverse_of_hasOtherProblemSymptom"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#File_Problem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasCD_DVD_ROMSymptom">
  <rdfs:domain rdf:resource="#CD_DVD_ROM_Problem_Symptom"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasCD_DVD_ROMSymptom"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="inverse_of_hasHardwareSymptom"/>
  </rdfs:subPropertyOf>

```

```

    <rdfs:range rdf:resource="#CD_DVD_ROM_Problem"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="inverse_of_hasMouseSymptom">
    <rdfs:domain rdf:resource="#Mouse_Problem_Symptom"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="hasMouseSymptom"/>
    </owl:inverseOf>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom"/>
    </rdfs:subPropertyOf>
    <rdfs:range rdf:resource="#Mouse_Problem"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasMonitorSymptom">
    <rdfs:range rdf:resource="#Monitor_Problem_Symptom"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#hasHardwareSymptom"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="inverse_of_hasMonitorSymptom"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Monitor_Problem"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasMouseSymptom">
    <rdfs:range rdf:resource="#Mouse_Problem_Symptom"/>
    <owl:inverseOf rdf:resource="#inverse_of_hasMouseSymptom"/>
    <rdfs:domain rdf:resource="#Mouse_Problem"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#hasHardwareSymptom"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="inverse_of_hasPC_AccountSymptom">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="hasPC_AccountSymptom"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#PC_Account_Issue"/>
    <rdfs:domain rdf:resource="#PC_Account_Issue_Symptom"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:ID="inverse_of_hasIT_AdministrationSymptom"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="inverse_of_hasEnterpriseWebsiteSymptom">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="hasEnterpriseWebsiteSymptom"/>
    </owl:inverseOf>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#inverse_of_hasOtherProblemSymptom"/>
    </rdfs:subPropertyOf>
    <rdfs:domain rdf:resource="#Enterprise_Website_Symptom"/>
    <rdfs:range rdf:resource="#Enterprise-Website_Problem"/>
  </owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:about="#inverse_of_hasPrinterSymptom">
  <rdfs:domain rdf:resource="#Printer_Problem_Symptom"/>
  <rdfs:range rdf:resource="#Printer_Problem"/>
  <owl:inverseOf rdf:resource="#hasPrinterSymptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasFileSymptom">
  <rdfs:domain rdf:resource="#File_Problem"/>
  <rdfs:range rdf:resource="#File_Problem_Symptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasOtherProblemSymptom"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasPerformanceSymptom">
  <rdfs:range rdf:resource="#Software_Performance_Problem_Symptom"/>
  <rdfs:domain rdf:resource="#Performance_Problem"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasSoftwareSymptom"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasHardwareSymptom">
  <rdfs:domain rdf:resource="#Hardware_Problem"/>
  <rdfs:range rdf:resource="#Hardware_Problem_Symptom"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="hasSymptom"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasOtherProblemSymptom">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasSymptom"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Other_Problem_Symptom"/>
  <rdfs:domain rdf:resource="#Other_Problem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasEnterpriseWebsiteSymptom">
  <rdfs:subPropertyOf rdf:resource="#hasOtherProblemSymptom"/>
  <rdfs:range rdf:resource="#Enterprise_Website_Symptom"/>
  <rdfs:domain rdf:resource="#Enterprise-Website_Problem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasPC_AccountSymptom">
  <rdfs:domain rdf:resource="#PC_Account_Issue"/>
  <rdfs:range rdf:resource="#PC_Account_Issue_Symptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="hasIT_AdministrationSymptom"/>
  </rdfs:subPropertyOf>

```



```

</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverse_of_hasHard_Drive_Symptom">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Hard_Drive_Problem"/>
  <owl:inverseOf rdf:resource="#hasHard_Drive_Symptom"/>
  <rdfs:domain rdf:resource="#Hard_Drive_Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasSymptom">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#isSymptomOf"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Help_Desk_Enquiry"/>
  <rdfs:range rdf:resource="#Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverse_of_hasKeyboardSymptom">
  <rdfs:range rdf:resource="#Keyboard_Problem"/>
  <owl:inverseOf rdf:resource="#hasKeyboardSymptom"/>
  <rdfs:domain rdf:resource="#Keyboard_Problem_Symptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasRemoteServerSymptom">
  <rdfs:range rdf:resource="#Remote_Server_Problem"/>
  <owl:inverseOf rdf:resource="#hasRemoteServerSymptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasOtherProblemSymptom"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Remote_Server_Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasCD_DVD_ROMSymptom">
  <rdfs:domain rdf:resource="#CD_DVD_ROM_Problem"/>
  <rdfs:subPropertyOf rdf:resource="#hasHardwareSymptom"/>
  <owl:inverseOf rdf:resource="#inverse_of_hasCD_DVD_ROMSymptom"/>
  <rdfs:range rdf:resource="#CD_DVD_ROM_Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPaswordSymptom">
  <rdfs:range rdf:resource="#Password_Issue_Symptom"/>
  <rdfs:domain rdf:resource="#Password_Issue"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasIT_AdministrationSymptom"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverse_of_hasMonitorSymptom">
  <rdfs:domain rdf:resource="#Monitor_Problem_Symptom"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom"/>
  </rdfs:subPropertyOf>

```

```

<rdfs:range rdf:resource="#Monitor_Problem"/>
<owl:inverseOf rdf:resource="#hasMonitorSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasIT_AdministrationSymptom">
  <rdfs:range rdf:resource="#IT_Administration_Issue_Symptom"/>
  <rdfs:domain rdf:resource="#IT_Administration_Issue"/>
  <rdfs:subPropertyOf rdf:resource="#hasSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasPaswordSymptom">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#inverse_of_hasIT_AdministrationSymptom"/>
  </rdfs:subPropertyOf>
  <owl:inverseOf rdf:resource="#hasPaswordSymptom"/>
  <rdfs:domain rdf:resource="#Password_Issue_Symptom"/>
  <rdfs:range rdf:resource="#Password_Issue"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverse_of_hasIT_AdministrationSymptom">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#isSymptomOf"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#IT_Administration_Issue_Symptom"/>
  <rdfs:range rdf:resource="#IT_Administration_Issue"/>
  <owl:inverseOf rdf:resource="#hasIT_AdministrationSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasSoftwareSymptom">
  <rdfs:domain rdf:resource="#Software_Problem"/>
  <rdfs:subPropertyOf rdf:resource="#hasSymptom"/>
  <rdfs:range rdf:resource="#Software_Problem_Symptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isInstalledBy">
  <rdfs:domain rdf:resource="#Non_Standard_Problem"/>
  <rdfs:range rdf:resource="#Installor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isSymptomOf">
  <rdfs:range rdf:resource="#Help_Desk_Enquiry"/>
  <rdfs:domain rdf:resource="#Problem_Symptom"/>
  <owl:inverseOf rdf:resource="#hasSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEmail_AccountSymptom">
  <rdfs:range rdf:resource="#Email_Account_Issue_Symptom"/>
  <rdfs:domain rdf:resource="#Email_Account_Issue"/>
  <rdfs:subPropertyOf rdf:resource="#hasIT_AdministrationSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverse_of_hasHardwareSymptom">
  <rdfs:range rdf:resource="#Hardware_Problem"/>
  <rdfs:subPropertyOf rdf:resource="#isSymptomOf"/>
  <rdfs:domain rdf:resource="#Hardware_Problem_Symptom"/>
  <owl:inverseOf rdf:resource="#hasHardwareSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_hasEmail_AccountSymptom">

```

```

<rdfs:range rdf:resource="#Email_Account_Issue"/>
<rdfs:domain rdf:resource="#Email_Account_Issue_Symptom"/>
<rdfs:subPropertyOf rdf:resource="#inverse_of_hasIT_AdministrationSymptom"/>
<owl:inverseOf rdf:resource="#hasEmail_AccountSymptom"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverse_of_hasOtherProblemSymptom">
  <rdfs:domain rdf:resource="#Other_Problem_Symptom"/>
  <rdfs:range rdf:resource="#Other_Problem"/>
  <owl:inverseOf rdf:resource="#hasOtherProblemSymptom"/>
  <rdfs:subPropertyOf rdf:resource="#isSymptomOf"/>
</owl:ObjectProperty>
<Software_Performance_Problem_Symptom
rdf:ID="Software_Slow_Performance"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <MS_Office_Problem rdf:ID="MS_Access_Problem"/>
    <MS_Office_Problem rdf:ID="MS_Excel_Problem"/>
    <MS_Office_Problem rdf:ID="MS_Outlook_Problem"/>
    <MS_Office_Problem rdf:ID="MS_PowerPoint_Problem"/>
    <MS_Office_Problem rdf:ID="MS_Word_Problem"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<CD_DVD_ROM_Problem_Symptom rdf:ID="Rom_Cannot_Record"/>
<Password_Issue_Symptom rdf:ID="Change_Password"/>
<Email_Account_Issue_Symptom rdf:ID="Email_Account_Termination"/>
<Software_Performance_Problem_Symptom rdf:ID="Software_Frozen"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Email_Account_Issue_Symptom rdf:ID="Email_Account_Cannot_Login"/>
    <Email_Account_Issue_Symptom rdf:ID="Email_Account_Maintenance"/>
    <Email_Account_Issue_Symptom rdf:ID="Email_Account_Setup"/>
    <Email_Account_Issue_Symptom rdf:ID="Email_Account_Suspension"/>
    <Email_Account_Issue_Symptom rdf:about="#Email_Account_Termination"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Password_Issue_Symptom rdf:about="#Change_Password"/>
    <Password_Issue_Symptom rdf:ID="Invalid_Password"/>
    <Password_Issue_Symptom rdf:ID="Password_Syntax_Info"/>
    <Password_Issue_Symptom rdf:ID="Reset_Password"/>
    <Password_Issue_Symptom rdf:ID="Retrieve_Password"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Keyboard_Problem_Symptom rdf:ID="Keyboard_Long_Delay"/>
<Printer_Problem_Symptom rdf:ID="No_Printout"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <File_Problem_Symptom rdf:ID="File_Accidentally_Deleted"/>
    <File_Problem_Symptom rdf:ID="File_Accidentally_Modified"/>
  </owl:distinctMembers>
</owl:AllDifferent>

```

```

    <File_Problem_Symptom rdf:ID="File_Corrupted"/>
    <File_Problem_Symptom rdf:ID="Missing_File"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Mouse_Problem_Symptom rdf:ID="Mouse_Button_Not_Responding"/>
<Remote_Server_Problem_Symptom rdf:ID="Cannot_Login_to_Server"/>
<Mouse_Problem_Symptom rdf:ID="Cursor_Frozen"/>
<Keyboard_Problem_Symptom rdf:ID="Entire_Keyboard_Not_Responding"/>
<Enterprise_Website_Symptom rdf:ID="Website_Cannot_Completely_Load"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Hard_Drive_Problem_Symptom rdf:ID="Hard_Drive_Cannot_Shut_Down"/>
    <Hard_Drive_Problem_Symptom rdf:ID="Hard_Drive_Cannot_Start"/>
    <Hard_Drive_Problem_Symptom rdf:ID="Hard_Drive_Hung"/>
    <Hard_Drive_Problem_Symptom rdf:ID="Hard_Drive_Overheat"/>
    <Hard_Drive_Problem_Symptom
rdf:ID="Hard_Drive_Plugin_Not_Responding"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Enterprise_Website_Symptom rdf:ID="Website_Error_Message"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <CD_DVD_ROM_Problem_Symptom rdf:ID="ROM_Cannot_Close"/>
    <CD_DVD_ROM_Problem_Symptom rdf:ID="ROM_Cannot_Open"/>
    <CD_DVD_ROM_Problem_Symptom rdf:ID="ROM_Cannot_Play"/>
    <CD_DVD_ROM_Problem_Symptom rdf:about="#Rom_Cannot_Record"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Installor rdf:ID="Help_Desk"/>
<PC_Account_Issue_Symptom rdf:ID="PC_Account_Setup"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Installor rdf:about="#Help_Desk"/>
    <Installor rdf:ID="Vendor"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Remote_Server_Problem_Symptom rdf:ID="Missing_Folder"/>
<Monitor_Problem_Symptom rdf:ID="No_Image"/>
<Remote_Server_Problem_Symptom rdf:ID="Server_Slow"/>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Enterprise_Website_Symptom rdf:ID="Abnormal_Website>Loading_Speed"/>
    <Enterprise_Website_Symptom
rdf:about="#Website_Cannot_Completely_Load"/>
    <Enterprise_Website_Symptom rdf:about="#Website_Error_Message"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<Monitor_Problem_Symptom rdf:ID="Abnormal_Image"/>
<PC_Account_Issue_Symptom rdf:ID="PC_Account_Cannot_Login"/>
<owl:AllDifferent>

```

```

    <owl:distinctMembers rdf:parseType="Collection">
      <Software_Performance_Problem_Symptom
rdf:ID="Software_Abnormal_Performance"/>
      <Software_Performance_Problem_Symptom rdf:ID="Software_Cannot_Start"/>
      <Software_Performance_Problem_Symptom rdf:about="#Software_Frozen"/>
      <Software_Performance_Problem_Symptom
rdf:about="#Software_Slow_Performance"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <Mouse_Problem_Symptom rdf:ID="Mouse_Movement_Too_Slow"/>
  <PC_Account_Issue_Symptom rdf:ID="PC_Account_Termination"/>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <PC_Account_Issue_Symptom rdf:about="#PC_Account_Cannot_Login"/>
      <PC_Account_Issue_Symptom rdf:about="#PC_Account_Setup"/>
      <PC_Account_Issue_Symptom rdf:ID="PC_Account_Suspension"/>
      <PC_Account_Issue_Symptom rdf:about="#PC_Account_Termination"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <Mouse_Problem_Symptom rdf:ID="Mouse_Movement_Too_Fast"/>
  <Monitor_Problem_Symptom rdf:ID="Blackspot"/>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <Keyboard_Problem_Symptom
rdf:about="#Entire_Keyboard_Not_Responding"/>
      <Keyboard_Problem_Symptom rdf:about="#Keyboard_Long_Delay"/>
      <Keyboard_Problem_Symptom rdf:ID="Particular_Key_Not_Responding"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <Printer_Problem_Symptom rdf:ID="Abnormal_Printout"/>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <Monitor_Problem_Symptom rdf:about="#Abnormal_Image"/>
      <Monitor_Problem_Symptom rdf:about="#Blackspot"/>
      <Monitor_Problem_Symptom rdf:about="#No_Image"/>
      <Monitor_Problem_Symptom rdf:ID="Screen_Flipping"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <Remote_Server_Problem_Symptom rdf:about="#Cannot_Login_to_Server"/>
      <Remote_Server_Problem_Symptom rdf:about="#Missing_Folder"/>
      <Remote_Server_Problem_Symptom rdf:about="#Server_Slow"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
      <Printer_Problem_Symptom rdf:about="#Abnormal_Printout"/>
      <Printer_Problem_Symptom rdf:about="#No_Printout"/>
      <Printer_Problem_Symptom rdf:ID="Toner_Level_Low"/>
    </owl:distinctMembers>
  </owl:AllDifferent>

```

```

    </owl:distinctMembers>
  </owl:AllDifferent>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <MS_Office_Problem rdf:about="#MS_Access_Problem"/>
  </owl:distinctMembers>
</owl:AllDifferent>
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Mouse_Problem_Symptom rdf:about="#Cursor_Frozen"/>
    <Mouse_Problem_Symptom rdf:about="#Mouse_Button_Not_Responding"/>
    <Mouse_Problem_Symptom rdf:about="#Mouse_Movement_Too_Fast"/>
    <Mouse_Problem_Symptom rdf:about="#Mouse_Movement_Too_Slow"/>
  </owl:distinctMembers>
</owl:AllDifferent>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.1, Build 284)
http://protege.stanford.edu -->

```

UserSolutionUI.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.util.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Ontology browsing</title>
</head>
<body>
Help Desk Inquiry
<%
HashMap list = (HashMap) session.getAttribute("list");
if (list!=null){
Iterator iter = list.keySet().iterator();
Iterator itNext = list.keySet().iterator();
itNext.next();
Object nextKey = new Object();
%>
<form method="get" action="/KMS/SolutionRetrievalAgent">
<%
while (iter.hasNext()){
    Object keyTitle = iter.next();
    Hashtable ht = (Hashtable) list.get(keyTitle);
    if (ht!=null){
%>

<br/>
<%if (ht.keys().hasMoreElements()) { %>
<select
<% if (!iter.hasNext()) {%>
onChange="location=this.options[this.selectedIndex].value;">
<%} else {%><%} %>
<option></option>
<%
        Enumeration it = ht.keys();
        if (itNext.hasNext()) nextKey = itNext.next();
        if (it!=null) while (it.hasMoreElements()){
            Object key = it.nextElement();
            System.out.println(key+" "+nextKey );
%>
<option
<%if (nextKey.toString().equals(key.toString())) {%>
selected="selected"
<%} %>
value="/KMS/InterfaceSoftwareAgent?user=user&useraction=welcome&
concept=<%=ht.get(key)%>&ontology=<%=key %>"><%=ht.get(key)%></option>
<%} %>
</select>
```

```

<br/>
<%}} } %>
<%
String solution = request.getAttribute("solution").toString();
String problem = request.getAttribute("problem").toString();
if ((!solution.equals(""))&&(!problem.equals(""))){
%>
Solution:<br/><pre>
<%=request.getAttribute("solution").toString() %>
</pre>
<br/><br/><br/>
Problem:<br/><pre>
<%=request.getAttribute("problem").toString() %>
</pre>
<br/>
<input type="submit" name="useraction" value="Refresh"/>
</form>
<%} else{ %>
<input type="submit" name="useraction" value="Refresh"/>
</form>
Solution for this query is not available, please contact Help Desk at 1234567.
<%}} %>
</body>
</html>

```


web.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
  <distributable />
```

- <!--

To use non XDoclet filters, create a filters.xml file that contains the additional filters (eg Sitemesh) and place it in your project's merge dir. Don't include filter-mappings in this file, include them in a file called filter-mappings.xml and put that in the same directory.

-->

- <!--

To use non XDoclet filter-mappings, create a filter-mappings.xml file that contains the additional filter-mappings and place it in your project's merge dir.

-->

- <!--

To use non XDoclet listeners, create a listeners.xml file that contains the additional listeners and place it in your project's merge dir.

-->

```
- <servlet>
  <display-name>SolutionRetrievalAgent</display-name>
  <servlet-name>SolutionRetrievalAgent</servlet-name>
  <servlet-class>SolutionRetrievalAgent</servlet-class>
</servlet>
- <servlet>
  <display-name>InterfaceSoftwareAgent</display-name>
  <servlet-name>InterfaceSoftwareAgent</servlet-name>
  <servlet-class>InterfaceSoftwareAgent</servlet-class>
</servlet>
- <servlet>
  <display-name>SolutionStoringAgent</display-name>
  <servlet-name>SolutionStoringAgent</servlet-name>
  <servlet-class>SolutionStoringAgent</servlet-class>
</servlet>
```

- <!--

To use non XDoclet servlets, create a servlets.xml file that contains the additional servlets (eg Struts) and place it in your project's merge dir. Don't include servlet-mappings in this file, include them in a file called servlet-mappings.xml and put that in the same directory.

```

-->
- <servlet-mapping>
  <servlet-name>SolutionRetrievalAgent</servlet-name>
  <url-pattern>/SolutionRetrievalAgent</url-pattern>
</servlet-mapping>
- <servlet-mapping>
  <servlet-name>InterfaceSoftwareAgent</servlet-name>
  <url-pattern>/InterfaceSoftwareAgent</url-pattern>
</servlet-mapping>
- <servlet-mapping>
  <servlet-name>SolutionStoringAgent</servlet-name>
  <url-pattern>/SolutionStoringAgent</url-pattern>
</servlet-mapping>
- <!--

```

To specify mime mappings, create a file named mime-mappings.xml, put it in your project's mergedir.

Organize mime-mappings.xml following this DTD slice:

```
<!--ELEMENT mime-mapping (extension, mime-type)>
```

```

-->
- <!--

```

To specify error pages, create a file named error-pages.xml, put it in your project's mergedir.

Organize error-pages.xml following this DTD slice:

```
<!--ELEMENT error-page ((error-code | exception-type), location)>
```

```

-->
- <!--

```

To add taglibs by xml, create a file called taglibs.xml and place it in your merge dir.

```

-->
- <!--

```

To set up security settings for your web app, create a file named web-security.xml, put it in your project's mergedir.

Organize web-security.xml following this DTD slice:

```
<!--ELEMENT security-constraint (display-name?, web-resource-collection+, auth-
constraint?, user-data-constraint?)>
```

```
<!--ELEMENT web-resource-collection (web-resource-name, description?, url-
pattern*, http-method*)>
```

```
<!--ELEMENT web-resource-name (#PCDATA)>
```

```
<!--ELEMENT url-pattern (#PCDATA)>
```

```
<!--ELEMENT http-method (#PCDATA)>
```

```
<!--ELEMENT user-data-constraint (description?, transport-guarantee)>
```

```
<!--ELEMENT transport-guarantee (#PCDATA)>
```

```
<!--ELEMENT login-config (auth-method?, realm-name?, form-login-config?)>
```

```
<!ELEMENT auth-method (#PCDATA)>
<!ELEMENT realm-name (#PCDATA)>
<!ELEMENT form-login-config (form-login-page, form-error-page)>
<!ELEMENT form-login-page (#PCDATA)>
<!ELEMENT form-error-page (#PCDATA)>

-->
</web-app>
```