

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2003

A multi-agent architecture for internet distributed computing system

Rodelyn Reyes Samson

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Systems Architecture Commons](#)

Recommended Citation

Samson, Rodelyn Reyes, "A multi-agent architecture for internet distributed computing system" (2003). *Theses Digitization Project*. 2408.

<https://scholarworks.lib.csusb.edu/etd-project/2408>

This Thesis is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

A MULTI-AGENT ARCHITECTURE FOR INTERNET
DISTRIBUTED COMPUTING SYSTEM

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Rodelyn Reyes Samson
September 2003


A MULTI-AGENT ARCHITECTURE FOR INTERNET
DISTRIBUTED COMPUTING SYSTEM

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino

by
Rodelyn Reyes Samson

September 2003

Approved by:



Dr. Arturo I Concepcion, Chair, Computer Science

27 May 2003

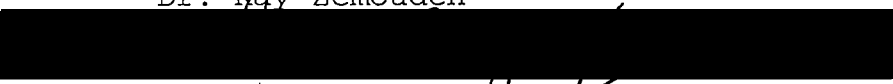
Date



Dr. Luis Sarmenta, Ateneo de Manila, Philippines



Dr. Kay Zemoudeh



Dr. Owen Murphy



© 2003 Rodelyn R. Samson

All Rights Reserved

ABSTRACT

This thesis presents the developed taxonomy of the agent-based distributed computing systems. Based on this taxonomy, a design, implementation, analysis and distribution protocol of a multi-agent architecture for Internet-based distributed computing system was developed.

The architecture was designed in a 100% pure agent system in a layered structure. Through this structure it is possible to redesign and reimplement a single module or layer without affecting the whole system (the main advantage of utilizing agents), which makes the system flexible and reusable. A prototype of the designed architecture was implemented on Spider III using the IBM Aglets software development kit (ASDK 2.0) and the language Java.

Through a hierarchical framework benchmarking, the performance measures of the multi-agent architecture in terms of agent overhead, network communication costs, speed up, and efficiency were analyzed. Then a distributed matrix multiplication was executed on Spider III, its execution time, speedup and distribution

efficiency running on the Internet were compared to Sequential and PVM programs. The results show that the Spider III multi-agent-based system succeeded in utilizing computer resources across multiple LANs on Abilene (Internet2) with various host architectures. For matrix multiplication problem, the Spider system has shown a better overall better performance than PVM and Sequential program in large matrix sizes. In small matrix size, Spider system shows a higher overhead due to serialization process and communication costs. In some tests, the bandwidth of the Internet, the high latency delay, the availability of the computer's idle processing power and the processor speed became the bottleneck of the system. The evaluation indicates that Spider III system is ideal for coarse grain application.

ACKNOWLEDGMENTS

As I approach the completion of this work, I cannot help but to look back and express my gratitude to those who have helped make all these possible.

Above all, I would like to give my deepest gratitude to God Almighty, whom without his guidance, blessings and love, this work would not have been possible.

Heartfelt thanks for my thesis advisor, Dr. A I Concepcion, for his continuous support and guidance to make this work possible. Your encouraging words, unending ideas, valuable advice and feedback kept me to find ways and solutions to what seems an impossible work. Thank you for believing in me especially in those times I struggled to continue.

I also would like to thank my thesis committee, Dr. Kay Zemoudeh, Dr. Owen Murphy and Dr. Luis Sarmenta. Thank you for the valuable advice and comment on my papers. This work has become more challenging and exciting with your sincere comments and feedback.

Thank you for the assistance of the staff of Department of Computer Science, CSUSB. I will always remember the time that Nam Kim and Ken have spent in

attending to all my calls and emails regarding network problem, computer authorizations, password lock-up, etc. Likewise, thank you to Monica Gonzales for helping me in registering my classes.

Thank you for the support of the members of Spider Team, Chunyan Ma, Li Wang, Rajeev Sadagopan and Jianhua Ruan. I owe a very special thanks to Jianhua Ruan, your continuous assistance and valuable advice not only as a co-partner on this work but as well as a good friend, will always be remembered.

Thank you to Dr. Yujun Wu of FNAL (Fermi National Accelerator Laboratory), Jameela Al-Jaroodi of UNL (University Nebraska-Lincoln) and Dr. Ernesto Gomez of CSUSB. I will always appreciate the time Dr. Yujun has spent in writing emails of encouragement and technical explanations about this work. For Dr. Gomez, thank you for explaining how I can run and test my MPI program in JB359.

Thank you for the participation of the following universities to use their computer laboratories, University of California Santa Barbara, University of California Riverside and University of Arizona. Through the assistance of Andy Pippin of UCSB, Mike Kennedy of

UCR and Rick Campbell of UA, I have been able to test the feasibility of running the Spider III on the Internet. Without your cooperation this work would not have been completed. As well as the award from Associated Students Inc. and National Science Foundation under the award 9810708 is gratefully acknowledged.

Thank you to some of my friends Kathy, Kirti, David Quintero, and Piyuthai. Although you have not directly contributed in my work, you have been part of my stay in CSUSB and made it easier and exciting, one way or the other.

A special thank you to Raymund Dahilig for willingly lending his laptop while I am working on this thesis, as well as to Shiela for staying at home to edit and proofread my work.

Thank you to my friends, Marlene, Ann, and Arlina. Thank you Marlene for constantly empowering my spirit and allowing me to work at night although you cannot sleep with the sound of my keyboard and lights on. Most of all thank you for all the lessons in life that I have learned from you that I would not have been able to learn even in any prestigious university in this world.

Thank you to all my friends and former schoolmates

back home who continuously gave moral support and inspiration in times of homesickness and loneliness.

Thank you to my grandma who continuously prays for my safety, likewise to my aunt Rosie and her family who have supported me from the very beginning of this program. She even learned how to use the library of UCR in borrowing books for me.

Most of all thank you to my family. To my dear parents Rodelio and Sonia Samson, who have been the sources of my strength and inspiration. Also to my sister and brother for your constant love, support, prayers and inspiration.

Finally, special thank you to my dear friend Richard Pangilinan who has been a constant source of inspiration to continue this work.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	v
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
CHAPTER ONE: INTRODUCTION	1
1.1 Related Works on Distributed Computing System	4
1.1.1 Beowulf	4
1.1.2 Mosix	5
1.1.3 Message Passing Interface	5
1.1.4 Parallel Virtual Machine	6
1.1.5 Java Parallel Virtual Machine	6
1.1.6 Bayanihan: Web-based Volunteer Computing	7
1.1.7 SETI@home	8
1.1.8 Spider	8
1.2 Thesis Goals	10
1.3 Limitations of the Thesis	12
1.4 Organization	13
CHAPTER TWO: OVERVIEW OF AN AGENT AND AGENT-BASED SYSTEMS	
2.1 Overview of an Agent	15

2.2 Agent-based Systems	18
2.3 Agent Design Paradigm	19
2.4 Components of Agent-based Systems	21
2.4.1 Agent	21
2.4.2 Worker Node	22
2.4.3 Server	22
2.4.4 Mobility	23
2.4.5 Communication	23
2.5 Survey of Commercial Mobile Agent Systems	25
2.5.1 Aglet	25
2.5.2 Toshiba's Agents	28
2.5.2.1 Bee-gent Framework	28
2.5.2.2 Plangent	29
2.5.3 Concordia	29
2.5.4 D'Agents	30
2.6 Taxonomy of Agent-based Systems	34
2.6.1 Classification of the Evaluated Mobile Agent Systems	37
2.7 Application Areas	38
CHAPTER THREE: INTERNET-BASED DISTRIBUTED COMPUTING SYSTEMS	
3.1 <u>SETI@home</u> and Bayanihan	41
3.1.1 SETI@home	42

3.1.1.1	Architecture	43
3.1.1.2	Design Issues of Distributed System	45
3.1.2	Bayanihan: Web-based Volunteer Computing	45
3.1.2.1	Architecture	46
3.1.2.2	Design Issues of Distributed System	47
3.2	Advantages of Agent-based Approach over Client/Server Approach	48
CHAPTER FOUR:	DESIGN OF THE MULTI-AGENT ARCHITECTURE	51
4.1	Architecture	52
4.1.1	Components of the Spider System ...	54
4.1.2	Important Design Issues of Distributed System	59
4.1.3	System Configuration	64
4.1.4	User APIs	65
4.1.4.1	Task Creation	65
4.1.4.2	Message Passing	67
4.1.4.3	Group Functions	68
4.1.4.4	Asynchronous Tasks	69
4.2	Comparison of the Architecture	69
4.2.1	Architecture	70

CHAPTER FIVE: PERFORMANCE EVALUATION AND ANALYSIS	
5.1 Objectives	76
5.2 Testing Environments	77
5.3 Performance Analysis Framework	78
5.3.1 Elements of the Multi-agent Architecture	79
5.3.2 Pattern of Interaction and Distribution Protocol	81
5.3.3 Benchmark to Measure the Performance of Multi-Agent Distributed Computing System	83
CHAPTER SIX: CONCLUSIONS AND FUTURE DIRECTIONS	
6.1 Conclusions	95
6.2 Future Directions	99
APPENDIX A: GLOSSARY	101
REFERENCES	106

LIST OF TABLES

Table 2.1	Properties of Agent	17
Table 2.2	Comparisons between Client/Server, Remote Evaluation, Code-on-Demand, and Mobile Agents	20
Table 2.3	Summary of the Surveyed Agents' Characteristics	27
Table 4.1	Comparison of the Architecture of <u>SETI@home</u> , Bayanihan and Spider Multi-agent System	72
Table 5.1	Measurement of Agent's Overhead conducted in 1 domain and in 4 domains ..	84
Table 5.2	Measurement of Communication Time using MPI and Agent	86
Table 5.3	Performance of Matrix Distributed Algorithm on the Internet2 (Abilene)	90

LIST OF FIGURES

Figure 2.1 Concepts of Agent-based Systems	33
Figure 2.2 Taxonomy of Agent-based Systems	35
Figure 3.1 The Architecture of SETI@home	44
Figure 3.2 The Architecture of Bayanihan	47
Figure 4.1 Components of Spider Host	55
Figure 4.2 An Example on how an Application is Distributed to the Spider system	58
Figure 4.3 Task Interface	66
Figure 5.1 Performance Model Approach	78
Figure 5.2 Comparison of Communication Time between PingPong in C-MPI and Java-Spider	87
Figure 5.3 Performance of Distributed Matrix Algorithm	92
Figure 5.4 Execution Time of Distributed Matrix Algorithm on the Internet2 (Abilene)	92
Figure 5.5 Speedup Performance of Spider Agent-based compared to Sequential on the Internet2 (Abilene)	93
Figure 5.6 Efficiency Measurement of Spider Agent-based compared to Sequential on the Internet2 (Abilene)	93

CHAPTER ONE

INTRODUCTION

The explosive development of networking, the proliferation of powerful workstations and the availability of platform-independent languages such as Java, led to a very active research in the area of distributed computing systems. Due to the huge amount of computers' idle processing resources left unutilized at most universities, research laboratories, and companies, there is a great advantage to be had on how these idle processing resources can be utilized to perform complex scientific computations, as opposed to using expensive supercomputer resources. The solution is to virtually integrate these idle processing resources in a scalable high performance distributed computing system.

This thesis aims to collect and utilize idle processing resources found on the Internet by designing and implementing an Internet-based distributed computing system using the new agent technology. This agent will propagate and migrate to other Local Area Networks (LAN) carrying code and programs to perform computations.

A distributed computing system (DCS) can be viewed as a virtual supercomputer machine that can solve a large computational problem. DCS is a collection of inexpensive computers that run their own operating system without having global memory or single clock, and each computer communicates by exchanging messages over a network.

In recent years, the use of heterogeneous collection of computers interconnected by one or more networks has become a widespread approach to high-performance computing. This approach solves a complex computational problem faster than a single powerful computer. By distributing the whole process into a collection of computers that gathers and combines the results later in the distributed computations. Systems such as SETI@home [30], Distributed.net [13], and Bayanihan [29] are good examples of distributed computing systems that used the Internet in collecting workstations to perform computation from different domains of the Internet. Although the results of several research projects and implementations are already impressive, these are only the beginning of the development and improvement of powerful distributed computing systems. Distributed

computing system have much to offer in the execution of applications in the areas of science, engineering, biology and economic systems.

In order to build and implement a powerful and efficient distributed computing system, the following properties should be considered in modeling and design of such systems:

1. *Transparency* is the hiding of the aspects of system's functionality from the user or programmer. It is considered the major advantage in the implementation of distributed system [7].
2. *Coherency* deals with the concept of the process of obtaining coherent but partial view of the system or a complete but incoherent view of the system, since there is no global clock [7].
3. *Fault tolerance* is the property of distributed system that recovers the whole process or partial process if one or more processors failed during program execution [7].
4. *Concurrency control* is the management issue of how to prevent one execution sequence from interfering with the others when they interleaved to avoid inconsistencies or give erroneous results [7].

5. *Heterogeneity* is providing the same kind of service for every kind of platform [7].
6. *Security* deals with the avoidance or tolerance of deliberate attacks. It refers to the techniques ensuring data in a network so they are not compromised [7].
7. *Scalability* refers to the ability of the system to adapt to an increased demand in the network. The system can be considered scalable if its resources can be expanded to accommodate greater computational power [7].

1.1 Related Works on Distributed Computing System

In this section, to better understand the concept of distributed systems the following systems: Beowulf, Mosix, Message Passing Interface, Parallel Virtual Machine, Java Parallel Virtual Machine, Bayanihan: Volunteer Computing, SET@home and Spider will be presented.

1.1.1 Beowulf

Beowulf [3] project was started in 1994 at the Center of Excellence in Space Data and Information Services (CESDIS). It is a cluster of parallel computers

that is considered an excellent platform to provide cost effective computing to be used in operational settings by computational scientists to be used as test-beds for system research and to be used as tools for teaching parallel programming courses in universities. It is a system that consists of one or more server nodes and one or more client nodes connected together via Ethernet and some other networking technology. Currently, Scyld Computing Corporation is hosting the Beowulf Project with the mission to develop and support Beowulf systems in larger commercial arenas.

1.1.2 Mosix

Mosix [27] is a software package that is specifically designed to enhance the Linux kernel with cluster computing capabilities. The core of MOSIX features load balancing, transparency, memory ushering, and file I/O optimization that responds to variations in the use of the cluster resources. MOSIX was developed at the Institute of Computer Science, Hebrew University of Jerusalem, Israel.

1.1.3 Message Passing Interface

MPI (Message Passing Interface) [19] is the library specification for message passing. It is proposed as a

standard for a broadly based committee of vendors, implementers and users. The main advantage of establishing message passing is portability and ease of use. Currently, MPI is managed by Mathematics and the Computer Division of the Office of Computational and Technology Research of the U.S. Department of Energy.

1.1.4 Parallel Virtual Machine

PVM (Parallel Virtual Machine) [18] is an integrated set of software tools and libraries that simulate general-purposes, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architectures. The main goal of PVM is to enable such a collection of computers to be used cooperatively for concurrent or parallel computing.

1.1.5 Java Parallel Virtual Machine

JPVM (Java Parallel Parallel Virtual Machine) [14] was developed as a network parallel computing software system with an additional goal of utilizing resources such as Macintosh and Windows-NT, based by using Java language. It is an enhanced PVM software system for explicit message passing based distributed memory (MIMD) parallel programming. However, it provides features such

as thread safety, multiple communication end-points per task, and default-case direct message passing not found in PVM.

1.1.6 Bayanihan: Web-based Volunteer Computing

Bayanihan [29] is an object-oriented computational platform for setting forced or true volunteer computing systems. This set-up was achieved by empowering a true spirit of *bayanihan* (a native language in the Philippines that means cooperation) from the computer users by volunteering their computer's idle processing power as simple as visiting a web site and downloading the applet running the application. The application runs as a background process or as a screen saver. This system succeeded in encouraging computers users to volunteer their processing power without worrying how they can join the computation compared to other systems where they have to face the burden of downloading, installing and configuring the system. In addition, by using HORB [20] (distributed object package that hides the details of network communications from programmers), it allows the programmer or the user to write parallel program or use the system without worrying the low-level details such as

synchronization, communication, load balancing etc. Dr. Luis F. G. Sarmenta developed Bayanihan as a Ph.D. dissertation in the MIT Laboratory for Computer Science, Massachusetts Institute of Technology.

1.1.7 SETI@home

SETI@home (Search for Extra Terrestrial Intelligence) [30] is a distributed system developed at the University of California Berkeley for searching massive amounts of radio telescope signals for signs of extraterrestrial intelligence. Like Bayanihan, this system encourages the computer user to volunteer their computer's idle processing power to be used in complex computations. Volunteers can participate in the computation process by downloading SETI@home software, which runs as a screen-saver or background process on the computer. SETI@home connects to the Internet only when it has to download or upload processed data from and into the SETI@home servers.

1.1.8 Spider

Spider is an on going distributed computing project in the Department of Computer Science at California State University San Bernardino (CSUSB). The name was derived by visualizing the spider as a distributed operating

system and the whole network as a web. Han-Sheng Yuh first proposed it as an object-oriented distributed system on his Master's thesis in 1997 [37]. It has been furthered developed by Koping Wang on his Master project [35], where he made improvements and implemented Spider II system. Spider I & II systems are developed using C++ that runs on Unix or Unix-like platforms. It was tested on SGI Indigo workstations and Irix operating system. RPC (remote procedure calls) and BSD sockets interface were used for the implementation of message-passing interface.

Spider I system consists of four major components: Task Manager, Registry Server, Object Server Broker, and Object-Servers. Spider II, which is presented in PDCS 2000 (Parallel Distributed Computing System Conference), improved and enhanced the existing features and functionalities of the Spider System. The communication protocol performance, load balancing, fault tolerance, and multi-tasking operation have been improved which optimized the performances and stability of the system. Mirrors for both Task Manager and Registry Server were implemented which will be activated if there are failures in the task manager or registry server.

Furthermore, NSF (National Science Foundation) has just awarded \$53 million to four U.S. research institutions to deploy a distributed terascale facility (DTF) [21]. DTF will be the largest, most comprehensive infrastructure ever deployed for scientific research and envisioned as the first infrastructure that will solve the most pressing scientific problems in the area of biology and genomics to astronomy.

1.2 Thesis Goals

The advancement of Internet technology, the availability of agent technology and the platform-independent language, Java, has motivated us to look for ways to improve the speedup gain, communication and distribution protocols, architecture, and scalability of distributed computing systems. Although, agent technology is still in the pre-mature stage and has not been tested to be useful in the area of distributed computing systems, this thesis is the first research attempt to study the advantages and disadvantages of agent programming model in implementing an Internet-based distributed computing environment. This thesis aims to utilize the computer's idle processing power distributed

over the Internet by presenting and building a distributed computational platform using the new agent technology. This computational platform will help scientists and research organizations to run their simulation models on the Internet. It will eliminate spending millions of dollars in buying supercomputing machines and will also maximize the utilization of the computer by using its idle processing power. In addition, it will also build cooperation from different organizations that are willing to donate or volunteer their computer idle processing power. We are not only going to build a super computing system but we will also promote cooperation, good relationship and interest for science and research. Furthermore, this system can also be used as a tool for professor in teaching distributed computing system, the students will easily understand the concept of scaling a huge distributed computing system.

To achieve the goals of this thesis the following will be considered:

1. Choosing an agent framework to provide the mechanism of dispatching, loading, re-tracing and executing agents on behalf of the user.

2. Design the architecture of an Internet-based distributed computing system using agents to achieve greater flexibility, fault tolerance, scalability and efficient load balancing.
3. Model the system functionalities, such as monitoring hosts, managing tasks, and inter-process communication through agents. Agents, which will carry algorithms and data, will also perform user tasks.
4. Implement the distributed computing system and run it on the Internet.
5. Test the performance and efficiency of the system, a performance evaluation through benchmarking will be conducted.

1.3 Limitations of the Thesis

Although, the implementation of the multi-agent architecture to Spider III has inherited the basic level security of the Aglets framework by limiting the task agents to access local file systems; security is a major concern that has not been fully addressed on this thesis. However, the work for security improvement is underway to solve the security issues of the multi-agent

architecture. In addition, the implementation of agent only support one-hop task distribution, it does not support multi-hop tasks distribution (migrated task can be migrated to another destination node), but interweaving the Aglet APIs and Spider APIs could achieve this goal. Lastly, this thesis only implemented a distributed matrix algorithm to validate the performance of the Spider III, no complex application has been established yet, but a simulation of a water movement model was implemented in Spider II and is on the way to test the simulation on the Internet.

1.4 Organization

This thesis is organized as follows: Chapter 2 presents an overview of an agent, agent technologies and agent paradigm. Also, the application areas of an agent, survey of commercial mobile agent systems as well as the developed taxonomy based on the presentation of the commercial mobile agent system will be presented. Chapter 3 presents the multi-agent architecture for Internet distributed computing systems, the design goals, the architecture and the distribution protocols. In addition, the architecture of SETI@home and Bayanihan compared with

Spider will be presented. In Chapter 4, performance evaluation of the system will be conducted through benchmarking to test its advantages and limitations. Lastly, the conclusion and future directions of the system will be discussed in Chapter 5.

CHAPTER TWO
OVERVIEW OF AN AGENT AND
AGENT-BASED SYSTEMS

This chapter provides an overview of an *agent*, which will show the reasons why we aim to model and implement a distributed computing system using the agent-paradigm. The agent technologies, concepts and paradigms will be introduced first, as well as some various research issues involved in implementing agent-based distributed computing systems. Then, a survey of commercial mobile agent systems, which triggered our minds to see that using an agent could be an efficient way to model distributed computing systems, will be presented. At the end, we will present the developed taxonomy of agent-based systems and some possible application areas of agent technologies.

2.1 Overview of an Agent

Researchers and developers define an *agent* in different ways. Some say it is a program or software, some say it is a paradigm or model, and others say it is an object. In searching for the exact definition of an *agent*, we followed the structure of Franklin [15] in

presenting *agent* according to its properties, characteristics and applications. Table 2.1 shows the list of properties and characteristics of an *agent*.

Based on the properties and characteristics of an agent, we acquired a clear idea that an *agent* can solve the inherent limitations of the conventional process-oriented and client-server approach, which also is stated in Messenger [17]. As defined, an *agent* can migrate from one location to another, and is capable of communicating and collaborating with other agents. Thus, computations based on agent generally have greater fault tolerance since objects can be dispatched to other functional nodes. An *agent* can also encapsulate complete algorithms; therefore, it narrows the semantic gap between distributed algorithms and implementations. Moreover, if an *agent* object will be implemented using the popular Java language, heterogeneous host architectures and operating systems can be used in scaling large distributed computing systems. Thus, an *agent* offers several potential performance advantages: it reduces the amount of network communication in the client-server model by having an agent with data, code and state

Table 2.1. Properties of Agent

Property	Other Names	Meaning
Reactive	Sensing and acting	Responds in a timely fashion to changes in the environment
Autonomous		Exercises control over its own actions.
Goal-oriented	Pro-active, purposeful.	Does not simply act in response to the environment.
Temporally continuous		Persistence of identity and state over long periods of time.
Collaborative		Can work in concert with other agents to achieve a common goal.
Communicative	Social	The ability to communicate with persons and other agents with language more resembling humanlike speech acts than typical symbol-level program-to-program protocols.
Adaptive	Learning	Able to learn and improve with experience.
Mobile		Able to migrate in a self-directed way from one machine to another.
Flexible		Actions are not scripted.

*Adapted, by permission, Franklin, S., and Grasser, A., *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*, Proc. of Third International Workshop on Agent, Theories, Architecture and Languages, Springer-Veslag, 1996.

migrate to the resources or host clients; it obtains inherent parallelism by propagating agents over the network and distributing the computations to heterogeneous architectures as long as the agent is implemented using Java and the host client has JVM (Java Virtual Machine).

2.2 Agent-based Systems

The *agent-based* system is considered to be a branch of the taxonomy of distributed computing [25]. According to Vigna [16], the *agent-based concept* is an extension of the earlier idea of *process migration, remote evaluation, code on demand, and mobile entities* (Definition will be found in the next section).

Although existing agent-based systems have been useful and effective in E-commerce, information retrieval, and mobile agent applications such as IBM Aglets [1], Plangent [34], Bee-gent [33], Concordia [9] and D'Agents[12], as of this writing, there is no existing systems that utilize the Internet technology using an agent-based approach in building a general-purpose, computational platform and high performance computing system. Systems such as SETI@home [30],

Distributed.net [13], and Bayanihan: Volunteer Computing [29] have succeeded in utilizing the Internet technology in scaling a computing system but have not been able to explore the world of agent technology.

2.3 Agent Design Paradigm

Design paradigm [6] according to Fugetta is essential for the design of the distributed applications. It provides guidance for the characterization of the location of the components before and after the execution of the service, and guidance for the computational component, which is responsible for the execution of code, and the location where the computation of the service actually takes place. Carzaniga [6] identifies remote evaluation, code on demand and mobile agent based on the client-server paradigm to exploit code mobility. Refer to Table 2.2. The *client-server* is a well-known and widely used paradigm. In the client/server paradigm, the client requests a service from the server who has the code and resources to execute the service, while *remote evaluation* and *code on demand* is a modification of the client/server paradigm. In remote evaluation, the code is moved from the client to the server at run time.

Table 2.2. Comparisons between Client/Server, Remote Evaluation, Code-on-Demand, and Mobile Agents

Paradigm	Content			
	Before		After	
	S _A	S _B	S _A	S _B
Client/Server	Part of Code A	Part of code resource B	Part of code A	Part of code resource B
Remote Evaluation	Code A	Resource B	A	Code Resource B
Code on Demand	Resource A	Code B	Resource Code A	B
Mobile Agent	Code A	Resource	—	Code Resource

*Adapted, by permission, Fugetta, A, Picco G. P., and Vigna G., *Understanding Code Mobility*, IEEE Transaction of Software Engineering, 24(5):342-61, May 1998.

The client has the code to execute the service, yet only the server has the resources to execute the service. In *code on demand*, the server maintains the code and the client loads the code as needed. The client has the resources and requests the code from the server. On the other hand, *mobile agent* represents a stronger turning away from the client-server paradigm. A *mobile agent* is a program, which includes data and code that is able to migrate to other machines during its lifetime. It is a mobile entity that autonomously migrates carrying code, data and state.

2.4 Components of Agent-based Systems

This section presents the components that make-up the agent-based systems such as agent, worker node, server, mobility and communication. These will be identified and discussed in detail to give a comprehensive knowledge of how these components are integrated in implementing an agent-based system.

2.4.1 Agent

An agent is an object or a program that performs some task (or set of tasks) on behalf of a person. The tasks include monitoring each computer usage, executing user defined algorithms and data in a remote computer, maintaining persistent state, and communicating with its owner, other agents or its environment. Each agent has its own thread of execution so that it can perform tasks of its own volition and has ability to transport itself from one agent-enabled host in a computer network to another. It consists of code, data, and its persistent state, which includes its program counter and frame stacks, or the agent values, where it can decide to continue at the destination agent system.

2.4.2 Worker Node

A *worker node* is a place that provides an environment for an agent to be created and to perform computation. It is a place where an agent can interact with the host and use its resources or available services. It is characterized by the virtual machine executing the agent's byte code (engine), its network address, its computing resources, and any services it may host.

2.4.3 Server

An *agent server* for some existing mobile agent systems creates, executes, transfers, and terminates agents. It has the main control of the agent and its behavior. We defined the *agent server* as a node where all worker nodes in a certain subnet are registered. It simply has the address list of the worker nodes that can be used by an agent in performing computations. This feature gives us greater fault tolerance in case the server crashes because another worker node can volunteer as a server node and the crash does not affect the rest of the system.

2.4.4 Mobility

One of the key features of an agent is its ability to move from one place to another at their own volition. In order for this to happen, the implementation of the language of the agent must provide a special statement or library call with the address of the destination place. When the call is executed, the agent is suspended, encoded for transmission, transmitted to the new place, decoded and installed to resume running at its new location.

There are basically two ways to support the mobility of agent as stated in the thesis [22]: *weak migration* and *strong migration*. In *strong migration*, the agent state includes all its data, as well as the execution state of its thread, which at the lowest level is represented by its execution context and call stack. While in *weak migration*, the agent has the ability to transfer code across different computational environments. Code may be accompanied by some initialization data, but no migration of the execution state is involved.

2.4.5 Communication

To do useful work, the agent must be able to communicate with other agents, hosts or worker nodes, and

their owners. The basic communication modes are *synchronous* and *asynchronous* communication discussed in [26]. In this thesis, we added a multicast mode in communicating with the number of agents performing tasks.

Cabri et al. [4] differentiated communication between agents into *direct* and *indirect*. *Direct* communication relies on message passing and is typically a client-server communication protocol. It is also called location-independent communication as stated in [11], where two or more agents communicate with one another without regard to any particular location in the network. While *indirect communication* implies that agents interact via blackboards located in each hosting environment, which is used as information spaces in which to store and receive messages locally, it is also called location-dependent communication [11], where two or more agents communicate with one another at a specific location in the distributed system. In order to communicate, either the sender or the receiver agent or both, must visit that specific location.

2.5 Survey of Commercial Mobile Agent Systems

This section presents the existing mobile agent systems, such as Aglets SDK [1], Plangent [34], Bee-gent Framework [33], Concordia [9], and D'Agents [12]. These systems will be evaluated and analyzed with respect to agent characteristics as discussed earlier in the beginning of the chapter. From these evaluations, taxonomy of agent-based systems will be presented.

To date, comparisons and surveys of mobile agent systems have focused mainly on the use of Java programming languages [23], agent-based technology concepts [22], and performance [31]. In this thesis, we focus on characteristics and behavior of agents for comparison, which is summarized in Table 2.3.

2.5.1 Aglet

The Aglet Software Development Kit (ASDK) [1] is the most commonly used platform for mobile agent systems developed by IBM Tokyo Research Laboratory in Japan. *Aglet* is the name of the mobile agent, which is written in Java language, and its migration is based on a proprietary Agent Transfer Protocol (ATP). The ASDK runtime consists of a visual agent manager, called

Tahiti, and the Aglet server, which can be installed on a HTTP-browser through the additional module software *Fiji*. This system provides a modular structure and an easy-to-use API for programming of aglets, facility for dealing with mobility and communication between agents, and has extensive support for security and agent communication. One of the appealing characteristics of Aglet is autonomy. Once it is created, it will decide where to go and what to do, and it possesses a temporally continuous state for its ability to live over a long period of time. However, based on the experiments conducted by Silva [31], *Aglets ASDK* was concluded as a robust platform in passing all the tests without crashing, but its performance was not good when compared to other platforms. Also, Silva has detected some garbage left in the memory of the Agencies that can lead to the deterioration of the performance of the application over time.

Table 2.3. Summary of the Surveyed Agents' Characteristics

System Name	Aglets	Bee-gent	Plangent	Concordia	D'Agents
Organization	IBM Corp	Toshiba	Toshiba	Mitsubishi	Darthmouth College
Platform	JDK 1.1/ higher ver	JDK 1.1	JDK 1.1	JDK1.1/ higher version	Unix
Languages	Java	Java	Java	Java	Tcl, Java, Scheme, Phyton
Limited Lifetime	No	No	No	No	No
Migration type	Weak	Weak	Strong	Weak	Strong
Migration mechanism	ATP, RMI	HTTP	HTTP	RMI, SS1	Sockets, email
Fault- tolerance	No	Yes	Yes	Yes	Yes
Detached computing	Yes	No	No	Yes	No
Reactive	No	No	Yes	Yes	Yes
Autonomous	Yes	Yes	Yes	Yes	Yes
Goal-oriented	Yes	Yes	Yes	Yes	Yes
Collaborative	Yes	Yes	Yes	Yes	Yes
Communicative	Yes	Yes	Yes	Yes	Yes
Adaptive	No	Yes	Yes	Yes	No
Mobile	Yes	Yes	Yes	Yes	Yes
Flexible	Yes	Yes	Yes	Yes	Yes

2.5.2 Toshiba's Agents

Toshiba has developed two agent systems: Bee-gent Framework [33] and Plangent [34]. Bee-gent is a communication framework in which the whole system is consists of agents, while Plangent is a software system consisting of intelligent agents performing actions on their own volition.

2.5.2.1 Bee-gent Framework. Bee-gent (Bonding & Encapsulation Enhancement Agent) is a 100% pure agent system developed in Toshiba, Japan [33]. This framework completely "Agentifies" the communication that takes place between software applications. The applications become agents, and agents carry all the messages. Thus, Bee-agent allows developers to build flexible open distributed systems that make optimal use of existing applications. This framework is comprised of two types of agents: "Agent Wrappers" is used to agentify existing applications and "Mediation Agents" which support inter-application coordination by handling all communications. Due to the characteristics of this system such as: ability of agents to transport messages (mobile), encapsulation of applications to agents (goal-oriented), adaptive response to the nature of requests to determine

the best course of action (adaptive), as well as ability to communicate and to collaborate with other agents in accomplishing a common goal (communicative), the framework has been made more interesting and flexible.

2.5.2.2 Plangent. Plangent is an intelligent agent system that performs tasks for human users developed in Toshiba at Japan [34]. The agent in the Plangent system (Plangent agent) has movement and planning capabilities. Through replanning, the agent searches for another way to achieve its goal if it fails to perform the given goal. It can move around the network, determine the best course of action in various situations, and act on itself. Thus, the agent working on this system is considered autonomous, mobile, reactive, goal-oriented, intelligent, and adaptive.

2.5.3 Concordia

Concordia [9] was first developed in 1997 at Mitsubishi Electric Information Technology Center America, (MEITCA). Like the IBM Aglet, it is a framework for developing and executing mobile agents written in Java, but it is mainly used in an electronic auction house. This system consists of a *server* for executing and transferring agents, and an *administration manager* for

remote server administration. It provides a rich set of features, like support for security, reliable transmission of agents, access to legacy applications, inter-agent communication, support for disconnected computing, remote administration, and agent debugging. Also, the agent in this system is autonomous and self-determined in its operation; it is unique since it is in control of its own itinerary. Although Concordia possesses the following characteristics: mobility, adaptation, collaboration, and communication, an experimental study conducted in Universidade de Coimbra [31] concluded that this platform is not very robust in situations of stress testing and it generates network traffic. It always gives an *OutOfMemory error* when running the benchmark on a second test with a big size agent (1 MB) and the garbage collection within the platform is also not done in an appropriate way. Thus it leads to a large deterioration in the execution of agents. These results show a major weakness in the performance of Concordia.

2.5.4 D'Agents

D'Agents [12] is a multi-language agent system under development at Dartmouth College Hanover, Massachusetts.

The first version was done in 1994 (based on its own script language) and in 1995 (using Tcl under the project name TIAS). It consists of four levels: the lowest level is an interface with transport mechanism, the second level is a server that runs on each machine, the third level consists of the execution environments (one for each supported agent language), and the top level is the agents themselves, which execute in the interpreters and use the facilities provided by the server. The main focus in the design of this system is the modularization to facilitate experimentation. Through its layered architecture, it is possible to redesign and reimplement a single module without affecting any other module. This makes D'Agent flexible and reusable. Aside from flexibility, the architecture has extensive support in the areas of dependability, security, and fault-tolerance. Furthermore, similar to the previous existing mobile agent systems, the agents on this system is autonomous and can move on their own volition to a given list of destinations, as well as communicate and collaborate with other agents in order to accomplish a common task. Based on the comparison of the existing mobile agent systems, we concluded that Plangent and

Concordia have strong agent-characteristics such as fault-tolerance, detached computing, reactivity, autonomy, goal-orientation, collaboration, communication, adaptation, mobility and flexibility. However these characteristics cannot guarantee a robust and good performance of the system. This will still depend on the applications implemented in the system, the architecture of the system, the language used in implementing the application, the migration type, and communication and collaboration of agents.

Based on these conclusions, we developed taxonomy of agent-based systems that is useful in identifying issues that needed to be addressed by our proposed multi-agent architecture.

Figure 2.1 shows a tree-graph with concepts of an agent-based system, which is divided into the following categories: language-based, mobility, migration type, communication, and applications. The language-based gives the programmer or designer the option to use the appropriate language such as flexible object-oriented language (C++, Java, Tcl, etc) in building or implementing the system. The mobility of agent can be both static and mobile, or mobile alone.

Agent-based System

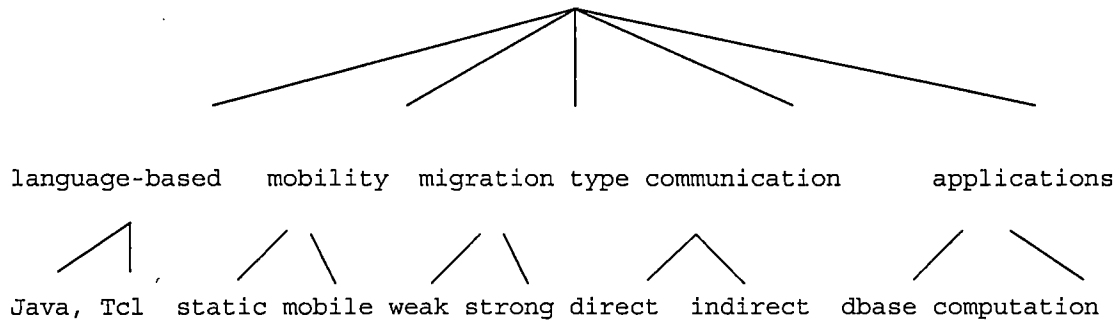


Figure 2.1 Concepts of Agent-based Systems

Static mobility is the characteristic of agent that does not move from one host to another, while a mobile agent has the ability to roam into network and communicates with other agents. In the migration type, strong migration is the ability of an agent that allow the migration of code, data, and the execution state to a different computational environment, while weak migration is the ability of an agent that allow code transfer across different computational environments. Code may be accompanied by some initialization data, but no migration of the state is involved. The next thing to consider is communication, which is one of the important concepts of agent-based systems. Communication could be classified into direct (location-independent) and indirect (location-dependent) communication. Direct communication

relies on message passing and is typically a client-server communication protocol. It is also called location-independent communication, where two or more agents communicate with one another without regard to any particular location in the network. Indirect communication implies that agents interact via blackboards located in each hosting environment, which is used as information spaces in which to store and receive messages locally. Indirect communication is also called location-dependent communication, where two or more agents communicate with one another at a specific location in the distributed system. In order to communicate, either the sender or the receiver agent or both, must visit the specific location. The concepts of an agent-based system will not be completed without considering the applications that will run in the system. The application could be a system or framework for a complex computation, database retrieval, network monitoring, commercial purposes, etc.

2.6 Taxonomy of Agent-based Systems

Based on the components and concepts of an agent-based system and the previous evaluation of existing

mobile agent systems, we developed taxonomy of an agent-based systems shown in Figure 2.2. An agent could be classified as intelligent, mobile, or stationary. An intelligent agent has the following characteristics: It is reactive, autonomous, goal-oriented, collaborative, communicative, adaptive, mobile and flexible. While a mobile agent has the ability to move in the network or different execution environments without planning

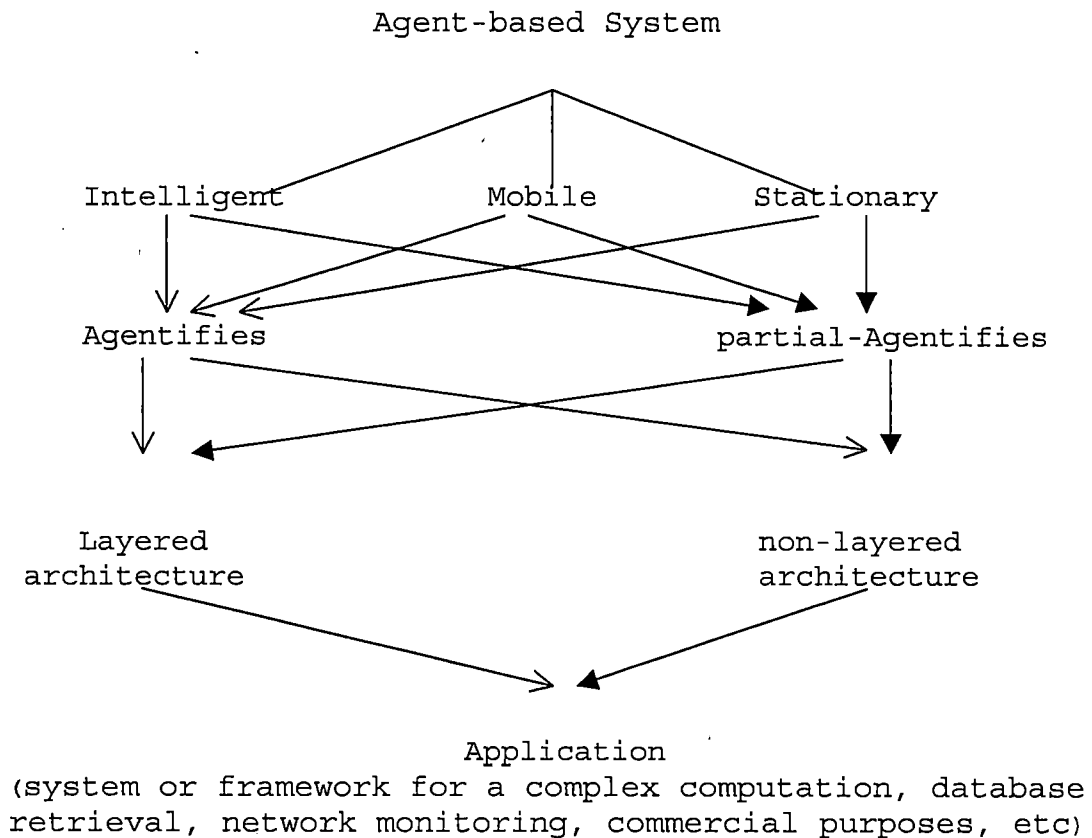


Figure 2.2 Taxonomy of Agent-based Systems

capabilities, neither reactive nor adaptive to any changes in the environment, this is agent is only autonomous, communicative and collaborative with other agents. Lastly, a stationary agent, as the name implies is static. It has no capability of moving or roaming in the network or to different places. This agent is only communicative. A good example is the stationary agent in a service provider system that communicates with a visiting mobile agent for any transaction or service. With these classifications of agents, an agent-based system is named "Agentifies" (a term borrowed from Toshiba's Bee-gent [32]) or "partial-Agentifies". An *Agentifies* system is a 100% pure agent in which the agents handle the application, the inter-process communication, the monitoring, the synchronization, and the load balancing of the system. While the *partial-agentifies* is a system in where the agent does not handle all the functions of the system, the agent can be responsible only for distributing or executing a task. The agents on *Agentifies* and *partial-Agentifies* systems can be intelligent, mobile or static. In addition, these systems can be implemented in a layered (modularization)

or non-layered approached which will help to determine the best application to integrate into the system.

2.6.1 Classification of the Evaluated Mobile Agent Systems

1. *Aglet SDK* is classified as *partial-agentifies* using a mobile agent. Its architecture was designed in modular object and the system was developed as a framework for mobile agent systems.
2. *Bee-gent* is classified as an *agentifies* system using mobile and intelligent agents. Its architecture was designed through modularization approach and the system was developed as a communication framework.
3. *Plangent* is classified as a *partial-agentifies* using intelligent agent. Its architecture was designed as a non-layered structure and the system was developed as a software system.
4. *Concordia* is classified as a *partial-agentifies* using mobile and intelligent agents. Its architecture was designed in non-lareyed structure and the system developed as a framework for mobile agent system.
5. *D'Agent* is classified as a *partial-agentifies* using mobile agent. Its architecture was designed in a,

layered structure and the system was developed as a framework for mobile agent system.

Based on the developed taxonomy of an agent-based systems, the proposed multi-agent architecture will be designed and implemented as *agentifies* system using mobile and static agents. The architecture will be implemented as a layered structure and the system will be used as a computational platform for distributed computing system.

2.7 Application Areas

This section presents a growing list of application areas for agent-based systems. Although mobile agents do not have a "killer application" yet, since everything that can be done with mobile agents can also be done in a more traditional way, a mobile agent system shows a promising approach to the development of distributed computing systems.

- *Mobile computing.* In a case where users do not want to stay online while performing computation or query, they can simply submit a mobile agent to perform the query, log off, and pick up the results later [21].

- *Internet information retrieval* [21]. Using an agent that moves to the place where data is actually stored, rather than moving all of the data through the network is a good approach for *data mining* [32].
- In *Electronic commerce* applications, users are enabled to perform business transactions through the network by an agent assisting user in negotiating [21].
- *Support and management of advanced telecommunications*. Agent can support a dynamic deployment of intelligent network service towards a real open programmable service environment [21].
- *Fault detection and monitoring large network*. In this case, the agents can monitor the system and bring the possible trouble spots to the attention of system administrator immediately [21].
- *Workflow applications*. The mobile agent supports the cooperation of persons and tools involved in a development process by representing activities as autonomous entities that are circulated among the co-workers in the workflow [21].

- *Parallel/Distributed Computation.* A computational model of mobile agent hosts is a feasible approach to distribute processes to multiple processors with a computation that requires huge processing power [21].

CHAPTER THREE
INTERNET-BASED DISTRIBUTED
COMPUTING SYSTEMS

In this chapter, we will present a two well-known Internet-based computing systems: SETI@home [30] and Bayanihan: Volunteer Computing System [29]. We will also discuss the architectures, functions and features of SETI@home and Bayanihan to show that the Internet can be a plausible way to build a powerful computing system. At the end, the advantages and motivations of using an agent-based approach compared to the traditional-approach in developing a distributed computing system will be presented.

3.1 SETI@home and Bayanihan

The continuous development of the Internet and the architecture of LAN have led to the utilization of idle CPU cycles distributed through the network of workstations to function as a powerful computing system comparable to the performance of an expensive single supercomputer. In fact, systems such as SETI@home [30] and Bayanihan [29] are good examples of Internet-based computing systems that utilize the massive computing

power of idle CPU cycles available through the Internet. To distinguish each system, their architecture, behavior and functionalities will be discussed in the first part of this section. Our intention of presenting existing Internet-based computing systems is to show that nowadays Web-based or Internet-based computing systems are possible with and comparable to a single and expensive supercomputing machine. We do not intend to overcome or surpass the existing systems, but we aim to present the concept that building an Internet-based distributed computing systems using mobile agents can be feasible and comparable in solving the burdens of traditional-based client-server paradigm.

3.1.1 SETI@home

SETI@home (Search for Extra-Terrestrial Intelligence) [30] was developed with the goals of collecting computers distributed over the Internet to be used in analyzing radio telescope data for signals from extra-terrestrial intelligence. This system is based on the "Divide and Conquer" paradigm wherein the huge sets of data are processed and divided into sub-processes and then distributed to other machines for further processing. SETI@home has even extended the usefulness of

the Divide and Conquer paradigm by using computers around the Internet to function as a parallel computing system. This set-up allows SETI@home to access more computing power than any existing supercomputer by encouraging computer users to share their idle processing power by downloading or uploading data. In fact it was stated in [23] that:

SETI@home is the largest distributed computation project in existence. It could also be considered to be the largest supercomputer in existence and the largest computation ever performed.

3.1.1.1 Architecture. Figure 3.1 shows the architecture of SETI@home, which is based on a traditional client/server approach. SETI@home collects data through a 305-meter radio telescope in Arecibo, Puerto Rico, and then records the data onto 35-Gbyte DLT tapes. The recorded data is shipped to Berkeley, to be subdivided into small work units on four splitters and transferred into temporary storage for distribution to volunteers. This system consists of three SUN Enterprise 450 series computers that function as science, data and user database servers. SETI@home currently distributes client software for 47 different combinations of CPU and operating systems. The program runs as a screen saver or

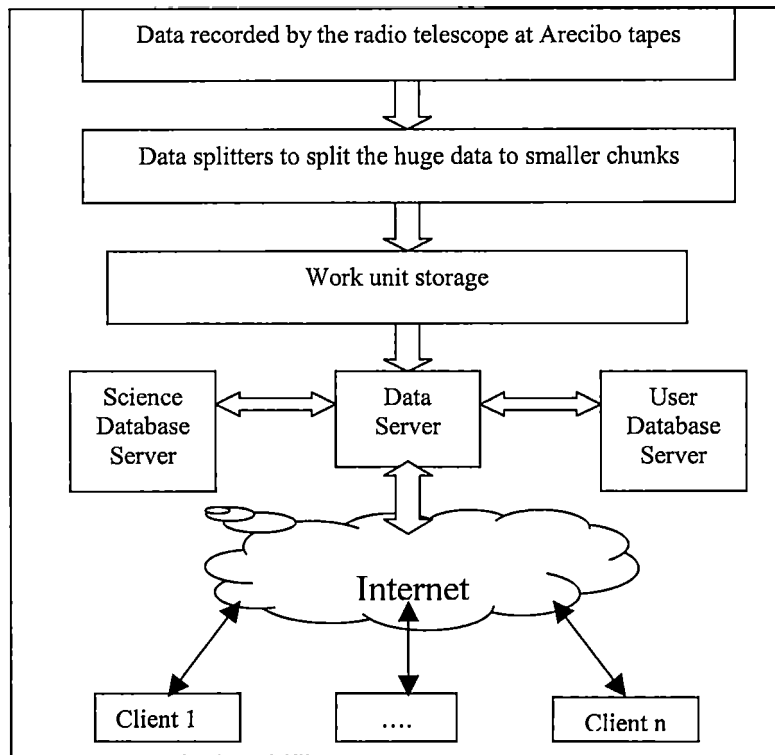


Figure 3.1 The Architecture of SETI@home
**Adapted, by permission, SETI@home: Search
 for Extraterrestrial Intelligence,
<http://seti@home.ssl.berkeley.edu>.*

as a background process (depending on the operating system) on the volunteer's machine and connects to the Internet only when it has to download data or upload processed data. Also, SETI@home supports a disconnected operation. Communication of SETI@home between the server and the client was achieved through hypertext transfer protocol (http). By using HTTP protocol, SETI@home allows those volunteers to go behind the firewall that prevents

access to the World Wide Web (WWW) in order to join the computation.

3.1.1.2 Design Issues of Distributed System.

SETI@home achieved fault-tolerance by checking errors made in the computation by examining the signals to determine if the parameters match their permitted values and by sending each work unit to multiple volunteers and cross-checking the returned values to verify accuracy. Meanwhile scalability was achieved by the ability to increase computational power from volunteers who had client software running as a screensavers or as a background process.

3.1.2 Bayanihan: Web-based Volunteer Computing

Bayanihan [29] is an object-oriented computational platform for setting a forced or true volunteer computing system. This set-up was achieved by utilizing a true spirit of *bayanihan* (a native language in the Philippines that means cooperation) from the computer users who volunteered their computer's idle processing power simply as visiting a web site and downloading the applet running the application. This set-up allows Bayanihan to easily scale large computing systems since the volunteers do not

encounter the burden of downloading, installing and configuring the system. In addition, Bayanihan used HORB [19] (distributed object package that hides the details of network communication from programmers), which allows the programmer or users to write parallel programs or use the system without worrying about the low-level details such as synchronization, communication, load balancing etc. Furthermore, Bayanihan is the first volunteer computing system that allows real-time control and supports interactive parallel computing.

3.1.2.1 Architecture. Like SETI@home, Bayanihan is based on a server/client paradigm. The server consists of a *manager*, *data pools*, and a *program object*. As shown in Figure 3.2, the *manager* distributes and collects data from its client engine and the *manager* has access to *data pools*. Then, the whole set of managers and data pools compose of a problem, which is contained in the *problem table*. From the *problem table*, the volunteers can choose which problem they want to compute. Each project in the Bayanihan system has a *program object*, which creates and controls managers, data pools, and data. On the client side, volunteers can participate in the computation using

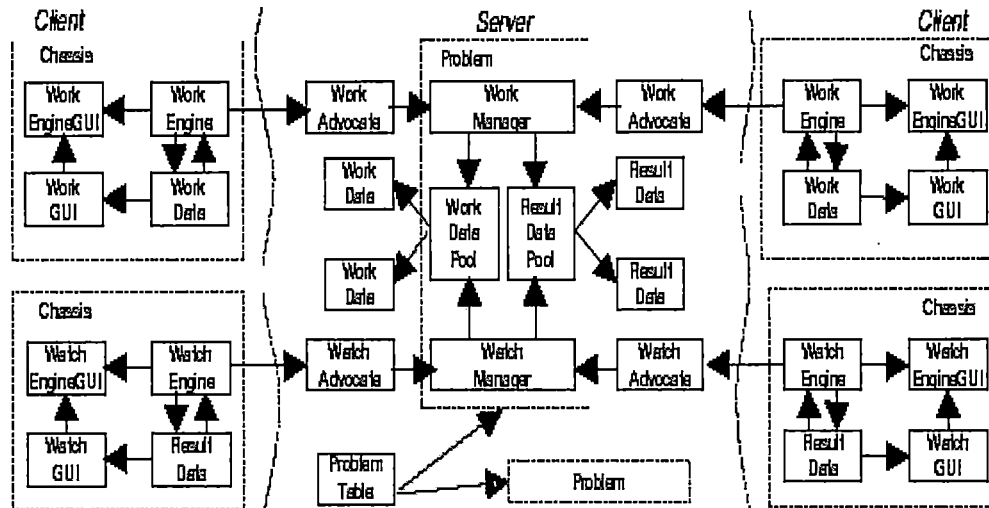


Figure 3.2 The Architecture of Bayanihan
 *Adapted, by permission, Sarmenta, L., Bayanihan:
Web-based Volunteer Computing, Ph.D. Dissertation.
 Dept. of Electrical Engineering and
 Computer Science, MIT, March 2001.

either the client from Web browser (Java applet), or the client from the command line (Java application).

3.1.2.2 Design Issues of Distributed System.

Bayanihan has achieved a fault-tolerant, scalable, flexible, transparent and adaptive parallel system. Through spot-checking, Bayanihan was able to detect the faulty node; once it was caught, the server backtracked through the results, recomputing any results depending on the offending node, and never allowed it to join the computation again. Scalability was achieved by developing a simple volunteer server, a set-up that bypasses the security restrictions of a Java-based volunteering

computing system. By using HORB (distributed object package similar to Sun's RMI), Bayanihan has achieved a transparent system that is able to hide the details of network communication, synchronization and load balancing from programmers and allows them to program in a fully object-oriented manner. Lastly, Bayanihan was modeled in an adaptively parallel [5] manner, which did not assume a fixed number of nodes, or depend on any static timing information; thus the system is flexible and scalable.

3.2 Advantages of Agent-based over Client/Server Approach

In this section, the advantages of an agent-based approach over the traditional client/server approach will be discussed. These are the important features of an agent-based computing model that make it attractive for constructing Internet applications.

1. The agent-based approach avoids transfer of large amount of data over the network. It is considerably cheaper to bring the processing method to the location where the data is stored ("function shipping") than to transfer the data to the client ("data shipping"). In [11] stated that, "one might argue that the same can be achieved through stored

procedures or similar mechanisms; while this is certainly correct, the power of the mobile agent approach is highlighted when proper processing methods are not known in advance but are data dependent." This approach offers two potential performance advantages: reduction of the amount of network communication in the client-server model by having an agent with data, code and state migrated to the resources or host clients, and the ability to obtain inherent parallelism by propagating agents over networks. Data mining [32] is the best example that will benefit from an agent-based system, in that the agent can migrate to the place carrying codes where a large amount of data is located and perform the necessary action.

2. The agent-based approach avoids any need for maintaining long-duration connections over the network. Since the transfer of code, data, and execution state into the computational environment is one of the agent-based systems' advantages, fixed connection to the network is unnecessary. This

provides a strong fault-tolerance feature so that a faulty node will not affect the whole system or computation. Computations based on agent generally have greater fault tolerance since objects can migrate or be dispatched to other functional nodes.

3. This approach promotes the construction of specialized servers by allowing a client's code that is tailor-made for clients' needs to be executed on the servers.
4. The agent-based approach provides a general-purpose, open framework for the development of distributed, networked systems, i.e., a universal computing platform.
5. It also supports for weak clients, a semantic routing, scalability, and reduced overhead for secure transactions, and comparatively robust remote interactions.
6. An agent can also encapsulate complete algorithms and thus narrow the semantic gap between distributed algorithms and their implementation.

CHAPTER FOUR
DESIGN OF THE MULTI-AGENT
ARCHITECTURE

Based on the developed taxonomy of agent-based systems in Chapter Two, we will design and implement the multi-agent architecture for Internet distributed computing system under the "Agentifies" concept using mobile and static agents in a layered (modularization) structure to be used as a computational platform.

In this chapter, we will present the architecture and protocol of the *Multi-agent Internet Distributed Computing System*. These architecture and protocol will show the feasibility of combining the Internet-based and agent-based concepts in building a powerful computing system. We will discuss the implementation of the multi-agent architecture to Spider III: Agent-based System. Then, we will present the load balancing, fault tolerance, scalability and system configuration of Spider III: Agent-based System. Last, to distinguish the different approaches we used in modeling and building the Spider multi-agent architecture, we will present a

comparison of the Spider architecture to SETI@home and Bayanihan.

4.1 Architecture

This section will present the design of a multi-agent architecture for Internet distributed computing system. The architecture uses mobile agents as the basic components for the transparent distribution of computations, and uses static and mobile agents to manage resource sharing, load balancing and fault tolerance of the system. To test the feasibility of the concept of multi-agent architecture, James Ruan [28] implemented the architecture; calling it Spider III agent-based distributed computing system. Detailed information can be found in the Master Project of Jianhua Ruan in 2002 [28] and in the paper presented in International PDCS '02 [8]. Spider is an on going distributed computing project in the Department of Computer Science at California State University, San Bernardino and the Spider version I and II have been discussed in detailed in section 1.1.8.

The agent-based approach is considered to be a branch on the taxonomy of distributed computing systems [25]. It is based on the concepts of static agents,

mobile agents and places. Places provide an environment for an agent to be created, to perform or to execute computations. It is a place where an agent can interact with the host and use its resources or services available, and it is located at a single node of the underlying network.

An agent could be multi-threaded entities whose state, data and code can be transferred to a new place when agent migration takes place, and each agent is identified by a unique agent identifier. By using an agent in the implementation of distributed computing systems, the inherent limitations of process-oriented approach can be addressed in the sense that the agent reduces global communication costs by moving the computation to the data and easily distributing the complex computations onto several possibly heterogeneous hosts. However in this thesis, we will only explore the advantages of distributing complex computations (data and programs) into several heterogeneous hosts using agents. Some advantages of using agents such as moving code to data (applicable to data mining) will not be utilized in this work.

4.1.1 Components of the Spider System

The Spider system consists of server nodes, worker nodes, and a Finder. In Spider, a worker node is a place that provides an environment for executing local as well as migrated tasks. A server is a node where all worker nodes in a subnet are registered. Any node, generally the first started node, can be the server node. The roles of server node and worker node are interchangeable, which means that a server node can become a worker node if it is heavily loaded; and a worker node can become a server node when the server node crashes (this will be discussed in later section). On the other hand, all server nodes are required to register in a Finder, so they can communicate with each other across WANs. The Finder is simply an RMI object registered at any host, with which all servers know how to communicate.

Figure 4.1 presents the components of the Spider host to be installed on both worker nodes and server nodes in the network. It consists of three layers: agent execution environment, system agents, and user agents. The agent execution environment is made up of a Java runtime environment and an agent framework which provides the mechanisms of loading, dispatching, retracting, and

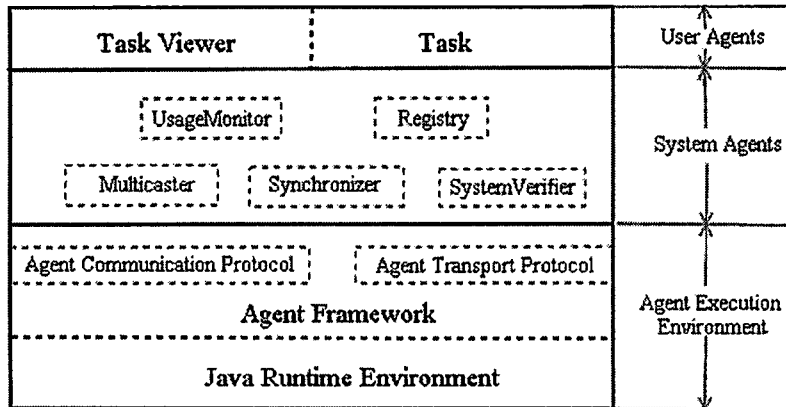


Figure 4.1 Components of Spider Host

executing agents on behalf of the user, and the environment supports communication between different agents. The system agent layer consists of Synchronizer, UsageMonitor, Registry, Multicaster, and SystemVerifier agents responsible for managing the system. Note that only Synchronizer and UsageMonitor agents start during startup time. Other components are passive and can be reactivated depending on the role of the node, and be deactivated when they are not being used. This layer also provides users with APIs to develop their own applications. The user agents layer consists of task viewer and tasks. Tasks are agents containing user defined algorithms and data, and are able to transparently migrate to other Spider hosts to execute.

Task viewer provides user interface to control and/or interact with tasks. A UsageMonitor agent runs on each worker node as well as on the server node. It measures the local CPU usage and updates its value to the Registry located on the server node. For a worker node, the usage determines whether it can lend computational power to other worker nodes; while for servers, the usage determines whether the server is overloaded so that another node can be selected to take over its task (this mechanism will be discussed in later section). The Synchronizer agent is also running on both worker node and server node for the purpose of fault tolerance. For a worker node, the Synchronizer detects whether a server is available; and for a server node, the Synchronizer detects whether there are multiple servers that co-exist in a local area network. The components of the server are Multicaster, Registry, and SystemVerifier. The Registry is located on the server to store addresses and usages of worker nodes, and waits to provide service to worker nodes that request a task distribution. Also, the Registry communicates with the Finder to update changes in the local subnet and request for available computing resources. The Multicaster and SystemVerifier work

together with the Synchronizer to ensure that system failures can be detected and recovered. Both worker nodes and server nodes provide the environment for executing local agents as well as visiting agents, so a task can be initiated from either place.

To process a work in Spider, Figure 4.2 illustrates an example of how an application is distributed to the Spider system.

1. User initiates a master task in a node through the Task Viewer. This task is executed locally without consulting the server;
2. The master task sends a request to the Registry agent for available worker nodes by remote inter-agent communication or by dispatching a messenger agent to the server;
3. The Registry replies with a list of addresses of available worker nodes; and
- 3'. If workers are not enough, the Registry communicates with the Finder and gets the location of other servers on the WAN. A list of addresses of available servers is also appended to the message to be sent back to the master task;

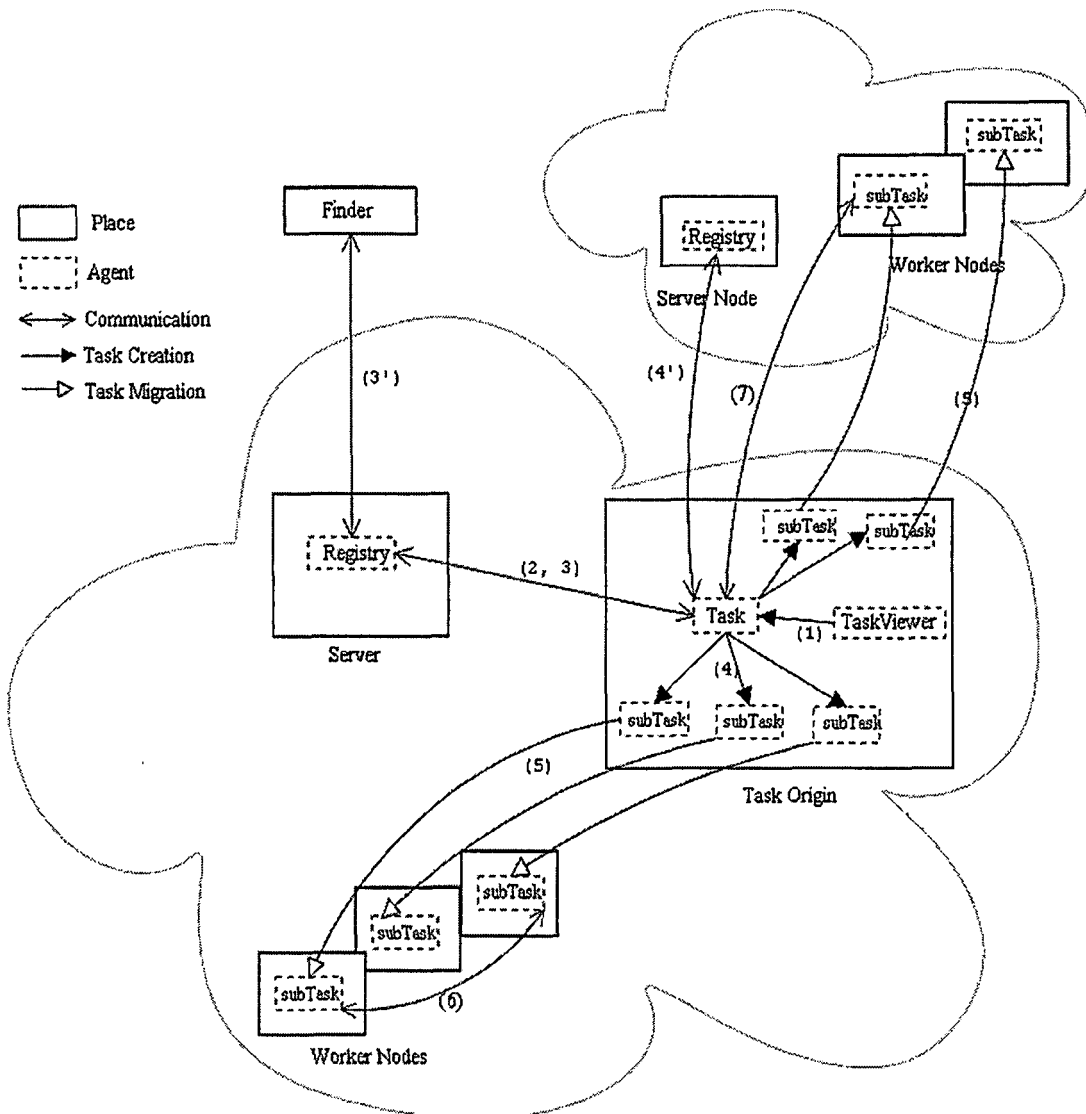


Figure 4.2 An Example on how an Application is Distributed to the Spider System

4. The master task spawns a subtask for each worker node;
- 4'. If the Registry replies with a list of server addresses, the master task talks with Registry agents on those servers to get more worker nodes;
5. Subtasks are migrated to worker nodes; A TaskID is returned to the master task for each spawned subtask. TaskIDs can be broadcast to subtasks so that each subtask knows about each other. Each subtask runs on its own host and can communicate with either other subtasks given that the TaskID is known.
6. Each subtask finishes its execution, sends the results back to the master task, and then dies. The master task collects the results from all subtasks and then dies.

4.1.2 Important Design Issues of Distributed System

James Ruan [28] implemented the multi-agent architecture, calling Spider III agent-based distributed computing system, on his Master Project to test the feasibility of the designed architecture. The implemented Spider III agent-based distributed computing system has

achieved load balancing, fault tolerance and scalability, which will be discussed on this section.

a. Load Balancing

Ruan [28] achieved Load balancing in each subnet by adapting the cyclic allocation mechanism used by PVM [17], with the addition of considering CPU utilization level as well as number of tasks. The Registry maintains the number of tasks running on each worker node and its CPU idle time. When a master task requests for spawning subtasks, it sends a message to the registry for a list of available worker nodes. A worker node will be considered available only when its CPU idle time satisfies a threshold value. Available worker nodes are further ordered according to the number of tasks running on the node. Nodes with the same number of tasks, are ordered by CPU idle time. Servers are informed of each other across WANs by the Finder. The load balancing of the worker nodes in different WANs; however, is more difficult to achieve because there is no global order of all worker nodes. One possible solution is to register the loading level of each subnet in the Finder. For example, the average number of tasks running on each worker node could be used. When the Registry receives a

request and the workload in the local subnet is heavier (i.e., average number of tasks is larger) compared to other subnets, then the request will be forwarded to a Registry where the workload is lighter.

The Spider system does not provide dynamic load balancing. A task migrated to a worker node cannot be migrated again. This is due to the fact that the Java Virtual Machine on which most agent frameworks runs does not support capturing the state of a thread, which would be a prerequisite for capturing and transferring execution state.

b. Fault Tolerance Feature

Ruan [28] classified system's failure into nodes failure and network failure. Node failure can be detected but is not distinguishable from network failure. We assume that nodes only suffer from crashes, which are a recovery failure. This type of failure causes the node to halt and lose its internal volatile state [2]. Also we assume that networks are fully connected; i.e., any node can talk to any other node directly. Detection and recovery of server failure is achieved with the cooperation of the Multicaster and the Synchronizer agent. The Multicaster agent multicasts the addresses of

the servers using a multicast sockets, while the Synchronizer agent checks multicast message to detect whether one or more servers are available. If a worker node does not receive multicast messages for a certain amount of time, it considers that the server node has crashed and will promote itself to be a server node by starting the appropriate agents. If two or more worker nodes promote themselves at the same time, they will detect each other and will demote to worker nodes immediately. Then each worker node waits a random amount of time before promoting itself, making it less likely for two worker nodes to promote themselves simultaneously. Once a new server is selected, all worker nodes will register to the Registry. Note that the information stored in Registry is only the address and usage of each worker node so the node can be fully restored after a new server is selected. Furthermore, the SystemVerifier on the server periodically checks that all registered nodes as well as the Finder are actually alive. A worker node is removed from the registry after being unreachable for a certain amount of time. Note that a faulty worker node can also be detected when a task attempts to migrate to that node. In that case, the

Registry will also need to be notified, thus reducing the frequency of running the SystemVerifier, which is costly. Similar to the SystemVerifier, the Finder uses a verifier thread to check whether all registered servers work properly. Besides, a newly selected server node can notify the Finder that the old server has been replaced. A communication failure with a Registry is also a hint that the server has already crashed.

Because the Java Virtual Machine does not allow capturing execution state, there is no way to completely restore a failed task; i.e., it is impossible to provide failure recovery for tasks. However, it is feasible and reasonable to let the task register a TaskFailed event listener. The master task will be notified when a subtask fails and the programmer can specify what actions to take upon failure; for example, whether to abort all subtasks including the master task, or to restart the failed task.

c. Scalability

Spider III agent-based distributed computing system [28] achieved scalability by having two-level of registry within the agent server and finder. Agent server has registry agent that stores information about the

addresses and usage of each worker node within a certain subnet, while the finder stores the addresses of the agent server located in different LANs. A task submitted from any location can take advantage of computational power on other domain by sending a request to the Finder to get a list of available worker nodes on other domain. This set-up makes the Spider system scalable by supporting the application or computation with additional computing power on other domain if the resources from the local domain are not enough.

4.1.3 System Configuration

To use the Spider III system, a set of Spider hosts (the agent server as well as the necessary managing agents) should be running on all nodes of interest. The Spider host program is provided in the form of a Java class and has to be started manually. After a Spider has started, it waits for multicast message to see whether a server already exists. If a server exists already, each spider host registers as a worker node; otherwise it declares itself as a server. The TaskViewer program (a console program or a graphical interface) is used by users to interact with the Spider system to shutdown the process, create tasks, list running tasks, etc.

4.1.4 User APIs

The programming interface implemented by Ruan [28] for Spider III agent-based distributed computing system is intentionally similar to that of the widely used PVM system, but with syntax and semantics enhancement supported by Java and better matched to Java programming styles. The central interface through which most Spider applications interact with the system is the Task class, which is an agent itself. It provides functions to control the creation of additional tasks and to support communication among tasks. Users develop their own applications by extending the Task class. Users will need to provide the *doJob()* function which will be executed after a task is created. Users can also override the *onInit()* and the *onExit()* functions to define specific behaviors during the task initialization or termination period. The basic interface of Task is depicted in Figure 4.3

4.1.4.1 Task Creation. The *spawn()* method provides the function to create additional tasks. It takes a string parameter indicating the name of a subtask class, which must also be a valid subclass of the generic Task class. Optionally, the initial arguments and the number

```

public class Task {
    // methods to be overridden
    void onInit(Object arg);
    void onExit();
    void doJob();

    // identity
    public TaskID myTid();
    public TaskID parentTid();

    // send messages
    public void sendMessage(TaskID tid, Object mesg,
        String tag);
    public void multicast(Vector tids, Object mesg,
        String tag);

    // receive messages
    public Message recieveMessage(TaskID tid, String
        tag, long timeout);

    // create additional tasks
    public TaskID spawn(String className, Object arg);
    public Vector spawn(String className, int count,
        Vector args);

    // group functions
    public int joinGroup(String groupName);
    public void leaveGroup(String groupName);
    public TaskID getTid(String groupName, int rank);
    public int getGroupSize(String groupName);
    public void broadcast(String groupName, Object
        mesg, String tag);
    public barrier(String groupName, int count);

    // asynchronous tasks
    public void addListener(Listener l);
}

```

Figure 4.3 Task Interface

- Adapted by permission, Ruan, J., *Spider III: A Multi-Agent-Based Distributed Computing System*, Master Project. Department of Computer Science, California State University, San Bernardino, June 2002.

of tasks to be spawned can also be specified. The initial arguments are passed to the `onInit()` method, which users can override to define the behavior during task initialization. The underlying Spider system decides which worker node is to be chosen to spawn a child task. The ID of a newly spawned task is returned to the parent task upon successful creation. Alternatively, a number of the similar tasks can be started with different parameters, in which case a vector of IDs are returned.

4.1.4.2 Message Passing. Message passing in the Spider system is performed by calling the `sendMessage()` and `receiveMessage()` methods of the Task class. Unlike in PVM, there is no separate packing/unpacking operation needed for message passing. Any object that implements the `java.io.Serializable` interface can be sent to another task. To send messages, a caller needs to provide a valid task ID, without knowing where the task is physically running. An optional tag can be provided to label the message. The `multicast()` method broadcasts the message to all tasks specified in the vector except itself. The `receiveMessage()` method can perform either blocking or non-blocking receive, based on the timeout parameter provided. A negative number of timeout blocks the

receiving task until the message arrives and a positive number of timeout blocks the receiving task until the message arrives or a given number of milliseconds has passed, whichever comes first. Optionally a task ID and a tag can be provided to match the incoming message.

4.1.4.3 Group Functions. The Spider APIs support creation of dynamic task groups. A task can join or leave a group at any time. A named group is created the first time when *joinGroup()* method is called. *JoinGroup()* returns the rank of the task in the group. A task can join multiple groups. The method *getTid()* returns a task ID with a given group name and the rank of that task in the group. The function *getGroupSize()* returns the size of the group. The method *broadcast()* is similar to multicast but it uses the group name as a parameter rather than the task ID's. Calling *barrier()* method causes the task to be blocked until *count* members of the group have called the function. Similar to the barrier function of PVM, a count is a given group name and the rank required because with dynamic task groups, the system cannot know how many members are in a group [17].

4.1.4.4 Asynchronous Tasks. To utilize the advantage of multi-thread programming of Java language, the Spider APIs support asynchronous communication mechanisms, such as asynchronous message queues, allowing asynchronous processing of requests. For example, a `MessageListener` whose `onMessageArrival()` method is being overridden can be registered using the `addListener()` method so that a task can do its job while waiting for messages. Furthermore, `subTaskFailed()` method of `TaskStatusListener` allows the programmer to define the specific actions to take when a subtask fails and the `nodesAdded()` method of the `SystemListener` gives the programmer flexibility to write programs that can adapt to the dynamically changing system size.

4.2 Comparison of the Architecture

This section presents the comparison of the architecture of SETI@home, Bayanihan, and the Spider multi-agent systems. These computing systems have a common goal of collecting huge amounts of idle computing power distributed over the Internet to be used in performing complex computations. Although, they have similar objectives and goals, these systems were

implemented and built with different architectures and purposes. These differences will be discussed in this section, which shows the relevance of building a distributed computing system using the new agent-paradigm.

4.2.1 Architecture

The computational models of SETI@home, Bayanihan, and Spider are based on master/slave pattern, "divide and conquer" paradigm, wherein huge sets of data are divided into small chunks then distributed to other machines for processing. SETI@home is a dedicated distributing system for processing huge amounts of data for radio signals from extra-terrestrial intelligence, and it supports a disconnected operation, while the Bayanihan and Spider systems are general-purpose distributed computing environments where the user can only provides the algorithm of the application to be computed without knowing the underlying system's architecture and protocol. In addition, Bayanihan is a volunteer computing system that is easy to use by simply visiting a web site and downloading the application to be computed, it also allows a real-time control and supports interactive parallel computing. On the other hand, the Spider multi-

agent system is the first computing system that utilized an agent in collecting resources distributed over the Internet. Moreover, Spider was built and implemented in a layer approach, which makes the system flexible. Each layer is independent of each other. In the middle layer lie the system agents that perform functions of the Spider system such as monitoring hosts, managing tasks, and performing inter-process communication. This layer is comparable to the HORB of Bayanihan, which hides the details of network communication such as synchronization, communication and load balancing from the computer user or programmer. Data transmission in Spider is handled by the IBM's Aglet in the lower layer, which is the Agent Execution environment.

SETI@home, Bayanihan and Spider have achieved scalability, load balancing, fault tolerance and security in different approach depending on their architecture. There is no way to tell which system is better, for it solely depends on the purpose of the systems, its applications and the architectures. Table 4.1 shows the summary of the comparisons of SETI@home, Bayanihan and Spider.

Table 4.1. Comparison of the Architecture of SETI@home, Bayanihan, and Spider Multi-agent System

Category	Bayanihan	SETI@home	Spider
Architecture	Client/Server	Client/Server	Agent-based
Model	Master/Slave	Master/Slave	Master/Slave
Computing	General-purpose	Dedicated	General-purpose
Communication	HTTP	HTTP	ATP (Aglet)
Synchronization	Yes	Yes	Yes
Scalability	Yes	Yes	Yes
Heterogeneity	Yes	Yes	Yes
Security	Strong	Minimal	Minimal
Fault-tolerance	Yes	Yes	Yes
Disconnected operation	No	Yes	No

a. Server

As discussed in the previous section, Bayanihan system has a manager in the server that handles the distribution and collection of data from its client. Similarly, SETI@home has a data server that handles the distribution and collection of work units from the client. Compared to Spider system, the server node does not handle the distribution of data/work to the client and the collection of the results. It only stores the list of addresses and usages of the Spider host in the system. The task agents handle the distribution and

collection of data/work initiated by the user, while the other functions of the system are handled by the system agents. Unlike Bayanihan and SETI@home, Spider system was built without a central point of control (the main advantage of utilizing agents); all the components of the Spider system are agents. If the server node crashes, it will not affect the computation and execution of the tasks, since the registry in the server node only stores the address and usage of each worker node. These addresses can be fully recovered after all worker nodes have registered to the Registry agent located in the new selected server.

The architecture of Spider III has solved the inherent problem of client-server structure where the server handles most of the functionalities of the system such as managing distribution and collection of the process or data, tracking the progress of the worker, etc. We also achieved a fault-tolerant system through this structure by not affecting the whole process of the computation in case the server node crashes.

b. Client

The client of Bayanihan, SETI@home, and Spider performs computation as a background process on the

computer. These systems do not affect or interfere with the computer user for they only use the idle processing power of the computer. In SETI@home, the client connects to the Internet only when it has to download data and upload processed data, while the client in Bayanihan and Spider systems connects to the Internet or network when starting and performing computation. Although, SETI@home has succeeded in reducing the amount of network communication in the client-server model and avoided any need for maintaining long-duration connections over the network, it requires volunteer to download the files and install the software. This is in contrast to Bayanihan, which addresses the issue of not having technical knowledge from volunteers in downloading the files, installing it and running the software, as well as addressing the security issue of hesitation from the user to allow user machines to be used in computation for the fear of a Trojan horse or virus that would steal or destroy their data. On the other hand, Spider has a different approach of scaling a large-computing system. It does not require the computer user to download or upload the data or work. This process is handled by the task agent, which migrates to the machine that satisfies

a threshold value of idle processing power. It temporarily borrows the computer's idle processing cycles without the computer user's knowledge and sends the result back after the computations.

c. Application

Since Spider III is still in the development and evaluation stage, this system has no complex application established yet to be compared with SETI@home and Bayanihan. However, a simulation of a water movement model was implemented in Spider II and is on the way to test the simulation on the Internet using the Spider III multi-agent system. In addition, Spider III is stable and robust enough to scale a large distributed computing system using the Internet. This can be attributed to the Spider's features of fault-tolerance, load balancing, scalability, and synchronization.

CHAPTER FIVE
PERFORMANCE EVALUATION
AND ANALYSIS

In this section, we will present the performance evaluation and analysis of the multi-agent architecture through the hierarchical framework benchmarking. This approach will determine the performance evaluation of each layer of the architecture and will determine the advantages and limitations of the Spider multi agent system.

5.1 Objectives

The main goal for conducting performance evaluation is to study the performance issues of the multi-agent architecture in terms of agent overhead, network communication costs, speed up and applications. We are concerned with the factors that affect the performance of the agents. To begin, we will establish a framework to analyze the multi-agent architecture. Then, we will realize this framework as a hierarchy of benchmark. This benchmark will serve as the basis for testing and evaluating the system. The evaluation will be performed using the combined machines from the computer

laboratories of the computer science department at California State University, San Bernardino; University of California, Santa Barbara; University of California, Riverside; and University of Arizona. From the results of the experiments and evaluations, we expect to find out the system's bottlenecks that will make way for the improvement of the Spider multi-agent system.

5.2 Testing Environments

The following are the specific machines used from four different campuses: The three machines we used from University of California Riverside are Sun Microsystems Netra t1 440 MHz Sparc processor with 256 MB RAM per node and the nodes were connected using 100MB Ethernet. From University of Arizona, we used a Sun Microsystems Ultra 30 248 MHz Sparc processor with 128 MB RAM. From the University of California Santa Barbara, we used five machines with AMD Athlon XP 1.4GHz 256 MB RAM; one AMD Athlon 1.1 GHz 256 MB RAM; one AMD Athlon XP 900 MHz 256 MB RAM; one Intel Celeron 500 MHz 128 MB RAM and one Intel Celeron 400 MHz 128 MB RAM, each node is connected using 100 MB Ethernet. From the computer laboratories at California State University San Bernardino, we used nine

machines with Intel Pentium 4 1.77GHz 256 MB RAM, each node is connected using 100 MB Ethernet. And from the research laboratory at California State University San Bernardino, we used the Spider server with Intel Pentium 4 1.77 GHz 256 MB RAM. The four universities are connected and also are members of Abilene (Internet2) Backbone Network. The Abilene (Internet2) has a goal to accelerate the creation of tomorrow's Internet.

5.3 Performance Analysis Framework

Based on the layered architecture of Spider, a hierarchy analysis framework is the chosen approach to determine its basic performance properties. Figure 4.1 illustrates the model for hierarchy analysis framework. The first and second layer (system agent elements, and pattern of interaction and distribution protocol) will be

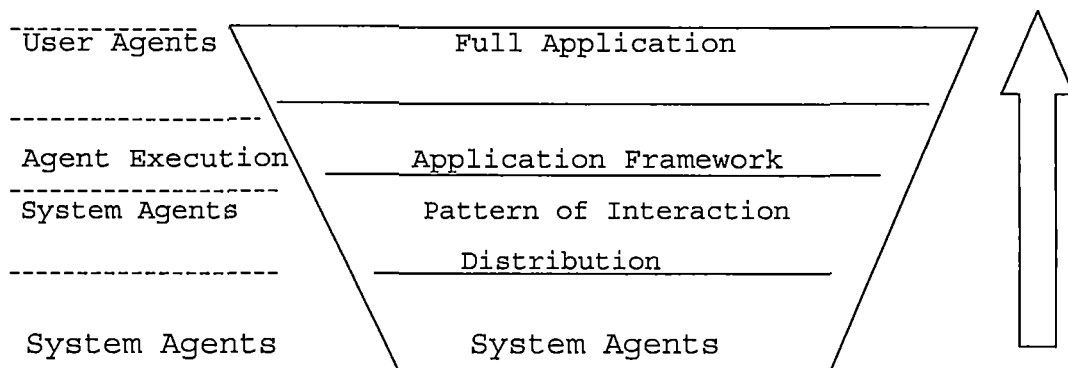


Figure 5.1 Performance Model Approach

the basis to test the agent overhead, while the third layer (application framework, resources, channel) will be the basis to test the communication cost of the system. The top layer (full application) will be the basis for testing the full application running in different domains.

The following are the procedures we performed in testing the performance of the Spider multi-agent system:

1. Identification of the elements of the multi-agent architecture such as system agents, server node, worker nodes (places), and behavior.
2. Identification of the pattern of interaction and distribution protocol.
3. Identification of the benchmark to measure the performance.
4. Building and testing the application framework based on the identified benchmark.

5.3.1 Elements of the Multi-agent Architecture

1. *System Agents* are composed of System Verifier, UsageMonitor, Synchronizer, Multicaster, and Registry.

a. *System Verifier* - periodically checks if all registered nodes and finder is alive and located on the server.

b. *UsageMonitor* - runs both on worker and server nodes, it measures the local CPU usage and updates its value to Registry agent in the server. For worker node, it determines whether it can lend computational power to other worker nodes, while for server, it determines whether the server is overloaded so that another node can be selected to take over.

c. *Synchronizer* - runs on worker and server node. For worker nodes, detects whether the server is available, and for the server, detects whether there are multiple servers that exist in a local area network.

d. *Multicaster* - an agent that multicasts the address of the server to worker nodes using multicast sockets.

e. *Registry* - stores addresses and usages of worker nodes and waits to provide service to worker nodes that request task distribution.

2. Worker node - a node in the Spider system, which has UsageMonitor and Synchronizer agents running.
3. Server node - a node in the Spider system where all the system agents (UsageMonitor, Synchronizer, SystemVerifier, Multicaster and Registry agents) run.
4. Finder - an object registered in any host, where all servers from different domain are registered.
5. Agent Execution environment - (IBM Aglet)
6. Task Agents - are agents migrating to worker nodes carrying data and execution codes.
7. Behavior - agents' behaviors are classified into creation, migration, execution, communication, synchronization and multicasting.

5.3.2 Pattern of Interaction and Distribution Protocol

The pattern of interaction and distribution protocol of agents is identified here to better understand how to initialize and run the application of the Spider System. Based on this interaction, a benchmark for performance evaluation has been generated (detailed discussion is in the next subsection).

1. Initialize/Launch the finder.

2. Run the Spider on several hosts (synchronizer and usage monitor are agents that will be loaded at this time).
3. One of the hosts will promote itself as a server. If there is a server running, all the worker nodes will register to the server through the registry agent.
(The remaining interaction and distribution protocol 4 -12 are shown in figure 3.4)
4. User initiates a master task in one of the node in a spider system through the task viewer. This task is executed locally without consulting the server.
5. The master task sends a request to the registry agent in the server.
6. The registry replies with the list of addresses of available worker nodes.
- 7.' If there are not enough workers, the registry talks with the finder and gets the location of the other servers on WAN.
7. The master task spawns a subtask for each worker node.
8. If the registry replies with a list of servers; then Subtasks are migrated to worker nodes.
9. A TaskID is returned to the master task for each

spawned subtask. TaskIDs can be broadcast to subtasks, each running on its own host, and can communicate to the other hosts given that the TaskId is known.

10. Each subtask finishes its execution, sends the results back to the master task, and then dies.
11. The master task collects the results from all subtasks and then dies.

5.3.3 Benchmark to Measure the Performance of Multi-Agent Distributed Computing System

1. Agent Overhead

To test the agent overhead, a matrix size of 300 * 300 was executed independently through the Sequential program using Java, and then through the agent. The average time was obtained for accurate results.

Agent overhead = $(Agent_{exe} - Seq_{exe}) / (Seq_{exe})$. From the results shown in Table 5.1, there was negligible or almost no overhead (0.93%) found in running the distributed matrix multiplication through agents in one domain compared to a sequential matrix multiplication. However, there was 12.5% overhead in running the application in several domains, due to the migration time of agent to other domains, limited bandwidth, high

Table 5.1. Measurement of Agent's Overhead conducted in 1 domain and in 4 domains

Run#	Sequential in Java	Agents in CSUSB	Agents in CSUSB, UA, UCR, UCSB
1	0.855	0.856	0.922
2	0.857	0.865	0.965
3	0.845	0.862	0.957
4	0.855	0.847	0.980
5	0.845	0.867	0.965
Average	0.8514	0.8594	0.9578
Overhead		0.93%	12.5%

CSUSB - collection of computers from CSUSB.

CSUSB, UA, UCR, UCSB - collection of computers from CSUSB, UCSB, UCR and University of Arizona

latencies and communication time of agent to the finder and to the registry agent of other domains. Based on the result of negligible overhead, we can justify that the agent-based paradigm is a better approach to distributed computing systems. Although, the tests show an overhead in running the application in several domains, it can be solved by the following approach: use a fast Internet connection; use the system in a coarse-grain application, where a higher computation to communication ratio is better; improve the interaction and distribution protocol specifically in communication time between agents; and improve the serialization and deserialization function of agents in sending objects from one processor to another,

since this process creates a communication overhead.

2. Communication Performance

To measure the communication cost, a ping-pong program was executed to exchange messages between two nodes. A ping-pong program is a common benchmark to measure the communication time between two nodes. It is used to measure the return trip time (RTT) for a message of a given size, and is written in an iterative way for messages of increasing size. Using the RTT, the effective bandwidth will be calculated from [20]. One program was written in C language using MPI (Message Passing Interface) and another was written in Java language for the Spider multi-agent system. These programs were executed in the Department of Computer Science at CSUSB.

$\text{Bandwidth} = (8 * \text{Message size} / 2^{20}) (\text{RTT} / 2)$. From the results in Table 5.2, it shows that the overhead in using C and standard MPI for small messages (less than two kilo bytes) makes the RTT (return trip time) almost constant. However, for larger message size (from four kilo bytes) the RTT grows proportional with the increase of message size. In agents, the overhead is initially higher due to agent serialization, but when the message size is larger (larger than 16 kilo bytes) the RTT grows proportional to

Table 5.2. Measurement of Communication Time using MPI and Agent

Message size (bytes)	Standard MPI in C language		Spider Multi-Agent In Java	
	RTT (sec.)	Bandwidth (Mbps)	RTT (sec.)	Bandwidth (Mbps.)
8	0.000087	1.4031	0.00675	0.0180
16	0.000088	2.7743	0.00675	0.0361
64	0.000088	11.0973	0.00677	0.1442
128	0.000099	19.7285	0.00678	0.2881
512	0.000170	45.9558	0.00678	1.1522
2048	0.000390	80.128	0.00681	4.5888
4096	0.000587	106.4735	0.01302	4.8003
16384	0.001633	153.0924	0.08455	2.9568
32768	0.003063	163.2386	0.08575	5.8309
65536	0.005896	169.6065	0.09254	10.8061
131072	0.011757	170.1114	0.14125	14.1593
250000	0.022362	170.5883	0.28150	13.5513
262144	0.023112	173.0703	0.28833	13.8729
524288	0.045779	174.7526	0.52510	15.2351
1048576	0.091220	175.4001	0.56951	28.0943
2097152	0.182710	175.1409	0.87225	36.6867
4194304	0.365523	175.0915	0.97654	65.5375

the increase of message size. It indicates that the overhead on agent communication affects the message with small sizes. The overall result shows that Spider agent-based system is ideal for coarse grain application where the higher computation to communication ratio has the better performance. In the future, if the overhead in the agent communication due to serialization is minimal,

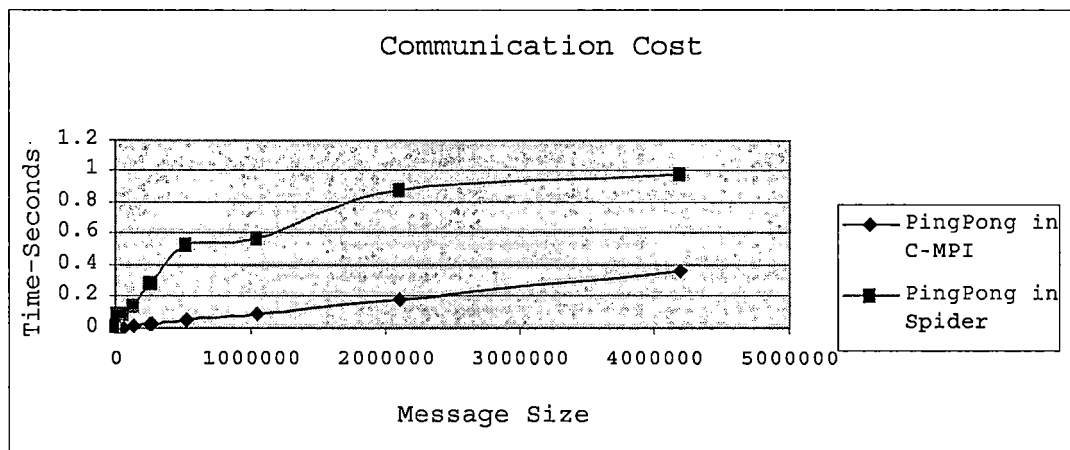


Figure 5.2 Comparison of Communication Time between PingPong in C-MPI and Java-Spider

Spider will be applicable to less coarse grain application.

3. Application

a. Comparison of PVM, Spider and Sequential performance in CSUSB domain

From the experiments Ruan has conducted in [7], Spider was compared to PVM in C and Sequential in Java in terms of speed up ratio by running a distributed matrix multiplication in the Department of Computer Science at California State University San Bernardino. Figure 5.2 shows the results of the experiments, where Spider has gained 0.9 efficiency compared to 0.7 efficiency of PVM. Speedup is measured by the execution time of the sequential algorithm to the execution time

of distributed algorithm, while efficiency is the percentage of the linear speed up and can be measured by the ratio, $\text{efficiency} = \text{speedup}/\text{number of nodes}$. From these experiments, the Spider agent-based system has the potential to improve the performance of distributed computing systems. To verify the consistency and efficiency of using agents, a distributed matrix multiplication with different sizes was executed in the collection of computers from CSUSB, UCR, UCSB, and University of Arizona. The tests were executed at 1:00 AM when most of the computers and network connection are idle. In running the tests, the fastest machines were used in 4 tasks and 9 tasks, and then a combination of fastest and slowest machines from UCSB, CSUSB, UA and UCR were used in running another 9 tasks with the same matrix sizes.

The results in Figure 5.3 show an increasing linear speed-up and efficiency in using the fastest machines in 4 and 9 tasks. Spider has gained 1.23 efficiency in running matrix size $1500 * 1500$ with 4-tasks and has gained 0.93 efficiency with 9-tasks. This is because of the following reasons: use of the fastest machines; increased size of data which takes longer for sequential

to finish the application due to the larger data sets to be processed that result in excessive cache misses and memory faults, compared to Spider where the whole task is divided into four processes that lead to fewer cache misses and memory faults. In addition, the bandwidth of the network is not a bottleneck at the time the tests are performed. Efficiency is much higher in 4 tasks since distribution, migration and collection of 9 tasks over the Internet creates a higher latency delay.

On the other hand, the results in running the application in the combination of fastest and slowest machines gave a negligible or insignificant efficiency due to 248 MHz and 440 MHz processors speed of the computers from UA and UCR respectively. On this set-up the slowest machines became the bottleneck of the computation. But it is worth mentioning that Spider started to gain speed up of 1.04 from 1200*1200 data.

The results indicates that the Spider agent-based and distributed matrix algorithm [41] is better than the sequential in larger data sizes even with the disadvantages set-up of using the slowest machines.

To check the validity of the efficiency of running 4 and 9 tasks, at 9:00 AM, a 1500 * 1500 matrix size was

Table 5.3. Performance of Matrix Distributed Algorithm on the Internet2 (Abilene)

Matrix Size	Task*	Time (sec)	Speedup	Efficiency
300 * 300	1	0.84	1.0	1.0
	4	1.05	0.8	0.2
	9	3.14	.27	0.03
	9*	7.451	0.11	0.01
500 * 500	1	4.25	1.0	1
	4	3.01	1.41	0.35
	9	5.16	0.82	0.09
	9*	6.48	0.66	0.06
600 * 600	1	7.75	1.0	1.0
	4	4.21	1.84	0.46
	9	7.11	1.09	0.12
	9*	21.92	0.35	0.06
900 * 900	1	30.21	1.0	1.0
	4	12.243	2.47	0.62
	9	13.07	2.31	0.26
	9*	44.48	0.68	0.06
1200 * 1200	1	89.39	1.0	1.0
	4	26.66	3.35	0.84
	9	23.11	3.87	0.43
	9*	85.75	1.04	0.12
1500 * 1500	1	242.59	1.0	1.0
	4	49.24	4.93	1.23
	9	28.86	8.41	0.93
	9*	238.45	1.02	0.12

4 and 9 tasks - running in the collection of fastest machines

9* tasks - running in the collection of fastest and slowest machines

executed. It gave a 0.85 efficiency compared to 1.23 efficiency obtained at 1:00 AM. At this time, the limited bandwidth of the Internet becomes the bottleneck of the efficiency, as well as the highest percentage of idle processing power. At 9 AM, most of the computer machines were utilized compared to the number being utilized at 1:00 AM.

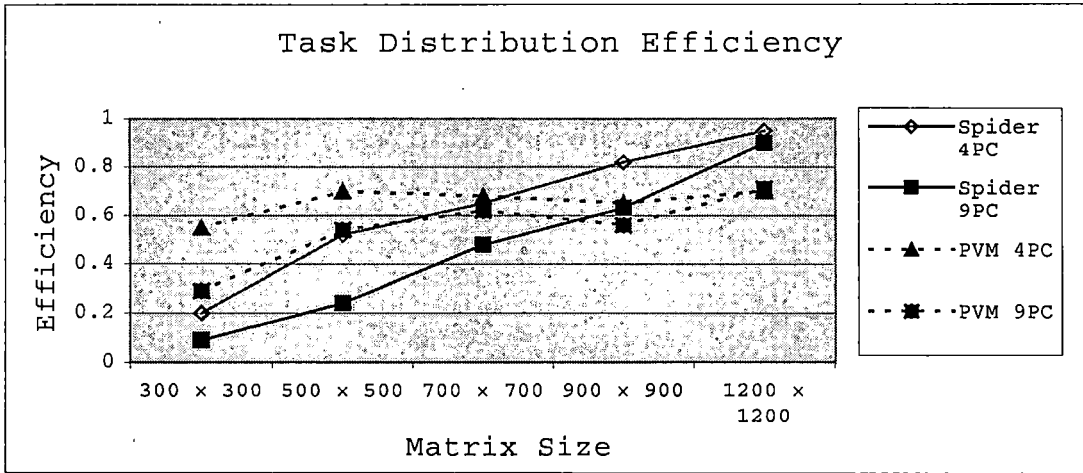


Figure 5.3 Performance of Distributed Matrix Algorithm

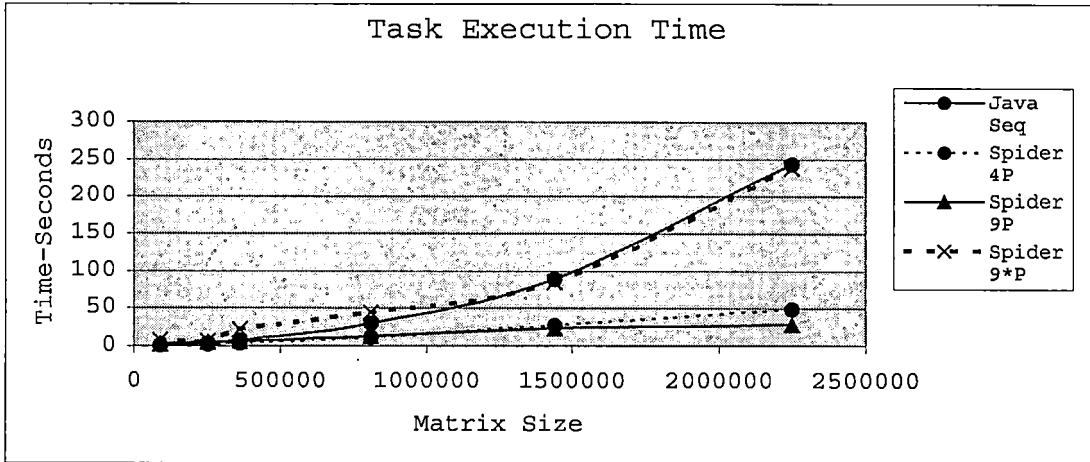


Figure 5.4 Execution Time of Distributed Matrix Algorithm on the Internet2 (Abilene)

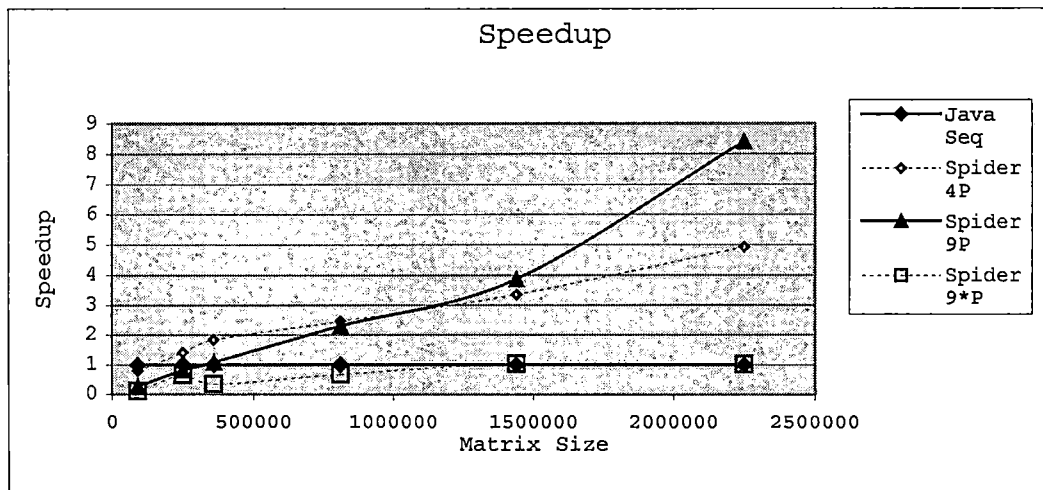


Figure 5.5 Speedup Performance of Spider Agent-based compared to Sequential on the Internet2 (Abilene)

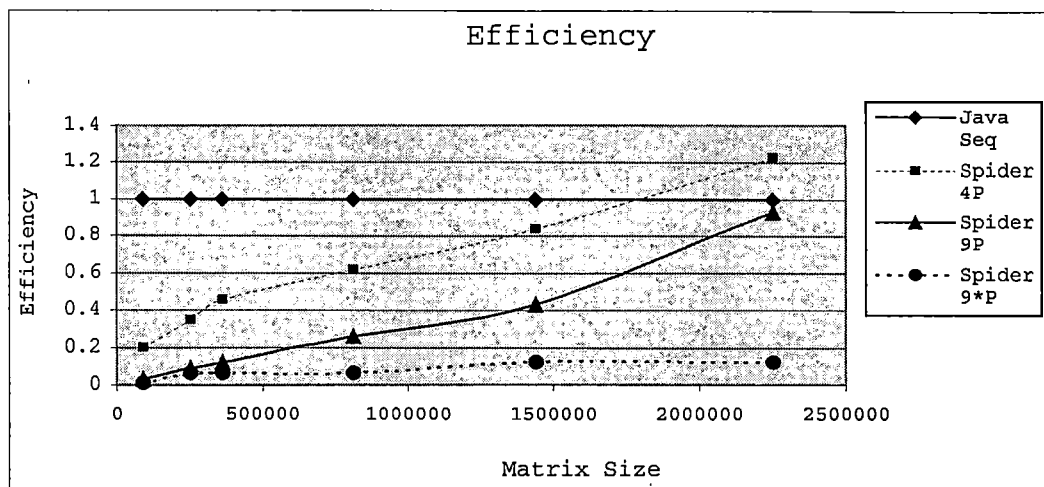


Figure 5.6 Efficiency Measurement of Spider Agent-based compared to Sequential on the Internet2 (Abilene)

In addition, from the result of the communication time (0.2815 sec) and execution time in 4-tasks (3.01 sec) for a 250,000 data, we validated the assumption of Ruan [28] on mathematical equation $t_{exe} > (1/n) * t_{load} + t_{mig} + t_{combine}$; that the performance of the agent-based model is better than the non-distributed solutions when the execution time of a subagent is greater than the sum of the average time to load an agent, the time to migrate an agent to a remote host, and the time to combine the results.

CHAPTER SIX
CONCLUSIONS AND FUTURE
DIRECTIONS

6.1 Conclusions

In this thesis, we envisioned utilizing the idle processing resources distributed over the Internet to scale-large computing systems by designing and implementing a multi-agent architecture for Internet-based distributed computing system. First, we presented the overview of an agent, the existing commercial mobile agent systems and the possible application areas of agent to gather an in-depth understanding and knowledge about agent.

Based on the presented overview, a comparison of five interesting mobile agent systems were compared and evaluated in terms of agents' characteristics, which subsequently led to the development of taxonomy of agent-based systems. Second, we presented two well-known Internet-based computing systems such as SETI@home [30] and Bayanihan [29], and we discussed each underlying architecture, functionalities and features. These systems were presented in order to show that the Internet could

be a plausible way to build powerful computing systems. Then we presented the architecture and distribution protocol of the multi-agent architecture for an Internet-based distributed computing system. We showed the feasibility of combining the Internet-based and agent-based concepts of building a large computing system by implementing the architecture and distribution protocol to Spider III. The IBM's Aglet software development kit (ASDK 2.0) [1] was used in providing the mechanisms of loading, dispatching, retracting, and executing agents on behalf of the user and supporting communication between different agents. The architecture was implemented in a layered-approach or modular object-oriented design, which provided separation of concerns and independence of the different components of agents.

After the presentation of Spider III: agent-based distributed computing system, we compared Spider III to SETI@home and Bayanihan in terms of the architecture of each system to show the relevance of building a distributed computing system using the new agent-paradigm. SETI@home and Bayanihan are based on a client/server paradigm, where the server handles the distribution and collection of data/work, unlike Spider,

which is built without a central point of control (main idea of utilizing agents). This set-up makes the Spider server crash tolerable because the server is dynamically selected from all participating nodes. In this case, we addressed some issues of fault-tolerance and removing the server in a bottleneck situation when it crashed.

Third, we developed a hierarchical framework benchmarking to analyze and evaluate the performance issues of the multi-agent architecture in terms of agent overhead, communication costs, speedup, and applications. A distributed matrix multiplication was programmed in Spider and its execution time, speed up and efficiency was compared to sequential program. The results show that using agent is a better approach for a distributed computing system and the Spider III: agent-based system performs well. We also validated the assumption of Ruan [28] on the mathematical equation $t_{exe} > (1/n) * t_{load} + t_{mig} + t_{combine}$; that the performance of our agent-based model is better than the non-distributed solutions when the execution time of a subagent is greater than the sum of the average time to load an agent, the time to migrate an agent to a remote host, and the time to combine the results. As of now, Spider III: multi-agent based

Internet distributed computing system is ideal for a coarse grain application where a higher computation to communication ratio is a better performance. In the future, if the overhead in the agent communication due to serialization is minimal, Spider will be applicable to less coarse grain application and fine grain parallelism.

This work demonstrated a computational platform that will provide researchers in any discipline such as biology, chemistry, or engineering the ability to run their simulation models on the Internet, as well the potential to eliminate spending millions of dollars in buying a supercomputing machine. In addition, this work will also build cooperation from different organizations that are willing to donate or volunteer their computers' idle processing power. We not only built a super computing system but also promoted cooperation, good relationships and increased interest for future science and research. Furthermore, this system can also be used as a tool for professors in teaching distributed computing systems; the students will easily understand the concept of scaling a huge distributed computing system. Lastly, we have proven that agent technology can provide flexible and powerful tools in managing and

coordinating resources, as well as increased distribution and performing complex computations.

6.2 Future Directions

Although, we achieved the objectives and goals that we aimed for in this thesis, this work has still many areas of research to be addressed such as:

- Providing support for detached/disconnected computing. This will eliminate maintaining long-duration connection to the Internet and provides a strong fault-tolerance feature; for example, a faulty node will not affect the whole system or computation.
- Enhancing the debugging tools and graphical user interface (GUI)
- Improving APIs to assist users in writing applications.
- Implementing a complex application such as running a Simulation of Water Movement Model.
- Establishing a mechanism where Spider system can be easily installed or run without asking special permission from the remote node intended to be used.

- Improving security so that agents will be trusted in the Internet2.

APPENDIX A

GLOSSARY

Agent	It is an object or program that performs some task (or set of tasks) in behalf of a person and has the ability to transport from one host to another host carrying code, data and state for execution.
Agentifies	A term used to classify distributed computing system developed with 100% agent. The inter-process communication, monitoring of the system, synchronization and load balancing are handled by agents.
Agent Server	A component of an agent framework that provides services to agent such as inter-agent communication, migration, execution, security, etc.
Aglet	It is the name of the agent of IBM's Aglet Software Development Kit
API	Application Programming Interface
ASDK	Aglets Software Development Kit
ATP	Agent Transport Protocol
Bayanihan	A Web-based Volunteer Computing system developed by Dr. Luis Sarmenta in MIT.

Bee-gent (Bonding & Encapsulation Enhancement Agent). A communication framework developed with 100% pure agent in Toshiba, Japan

CSUSB California State University, San Bernardino

Client/Server A well-known and widely used paradigm where the client requests a service from the service who has the code and resources to execute the service.

Concordia A framework for developing and executing mobile agent developed at Mitsubishi, America.

D'Agents A computational platform developed through agent at Dartmouth College, Massachusetts.

DTF Distributed Terascale Facility

GUI Graphical User Interface

JVM Java Virtual Machine

LAN Local Area Network

MESSENGERS An autonomous-object-based environment intended primarily for developing dynamically composed applications such as

	distributed simulations developed at UC, Irvine.
Mobile Agent	It is a mobile entity that autonomously migrate carrying code, data and state.
Multi-hop	A classification of migration on which a migrated task can be migrated to another destination node.
Node	A place that provides an environment for an agent to be created, and to perform computation. It is a place where an agent can interact with the host and use its resources or services available.
One-hop	A classification of migration on which a task can only migrate once.
Partial-Agentifies	A term used to classify a distributed computing system where agent does not handle all the functions of the system.
Plangent	An intelligent software system developed in Toshiba, Japan.
RPC	Remote Procedure Call
<u>SETI@home</u>	(Search for Extraterrestrial Intelligence) A dedicated distributed system for

searching extraterrestrial intelligence
signal developed at UC, Berkeley.

Speedup The ratio of sequential execution time to
parallel execution time.

Spider A distributed virtual machine system
developed in Department of Science, CSUSB.

UA University of Arizona

UCR University of California, Riverside

UCSB University of California, Santa Barbara

WAN Wide Area Network

REFERENCES

- [1] Aglets SDK Document, <http://aglets.sourceforge.net>
- [2] Aguilera, M.K., Chen, W., and Toueg, S., *Failure Detection and Consensus in The Crash-recovery Model*, Technical Report TR98-1676, Cornell University, Computer Science Department.
- [3] *Beowulf Introduction and Overview*, Scyld Computing Corporation, <http://www.beowulf.org/intro.html>.
- [4] Cabri, G., Leonardi, L., and Zambonelli, F., *The Impact of the Coordination Model in the Design of Mobile Agent Applications*, The 22nd Annual International Computer Software and Applications Conference (COMPSAC '98) (Vienna, Austria, August 19-21, 1998), August 1998.
- [5] Capello, P., Christiansen, B.O., Ionescu, M.F., Neary, M.O., Schauser, K.E., and Wu, D., *Javelin: Internet-based Parallel Computing Using Java*, *Proceeding of ACM Workshop on Java for Science and Engineering Computation*, June 1997. Also published in *Concurrency: Practice and Experience*, 9(11), Nov. 1997.

- [6] Carzaniga, A., Picco, G.P., and Vigna, G.
Designing Distributed Applications with Mobile Code Paradigms, 19th International Conference on Software Engineering (Boston, Massachusetts, May 17-23, 1997), ACM Press.
- [7] Chow, R., and Johnson, T., "Distributed Operating Systems & Algorithms," Univesity of Florida, Florida, US, 1997
- [8] Concepcion, A. I., Ruan, J., and Samson, R.,
Spider III: Agent-based Distributed Computing System, International Conference on Parallel and Distributed Computing System, Sept 2002, Kentucky, LA, USA.
- [9] *Concordia*, Mitsubishi Electric ITA, July 1999.
- [10] Cooper, T.E., Kedem, Z., and Rabin, M., *Parallel processing on networks of workstations: A fault-tolerant high performance approach*, in Proc. 15th IEEE International Conference on Distributed Computing Systems, 1995,
<http://cs.nyu.edu/milan/milan>.
- [11] *Crossroads, Technologies for the Development of Agent-Based Distributed Applications*, ACM Student

- Magazine *Parallel Computing*, Issue 8.3, pages 8-15, Spring 2002.
- [12] *D'Agents*. Dartmouth College, July 1999,
<http://agents.cs.dartmouth.edu/>.
- [13] Distributed.net. <http://www.distributed.net>
- [14] Ferrari, A.J., *JPVM: Network Parallel Computing in Java*, Technical Report CS-97-29, University of Virginia, Computer Science Department, Virginia.
- [15] Franklin, S., and Grasser, A., *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*, Proc. Of Third International Workshop on Agent, Theories, Architecture and Languages, Springer-Veslag, 1996.
- [16] Fugetta, A, Picco G. P., and Vigna G.,
Understanding Code Mobility, IEEE Transaction of Software Engineering, 24(5):342-61, May 1998.
- [17] Fukuda, M., *MESSENGERS: A Distributed Computing Based on Autonomous Objects*, Ph.D. Dissertation, University of California, Irvine, CA, 1997.
- [18] Geist, A., et. Al. *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press, 1994.

- [19] Herbert, S., *MPI: Message Passing Interface*, 1197,
<http://www.unix.mcs.anl.gov/mpi>.
- [20] Hirano, S., *HORB: Distributed Execution of Java Programs*, Proceeding of WWCA'97, volume 1274 of LNCS, pages 29-42, Springer, 1997,
<http://www.horbopen.org>.
- [21] <http://www-unix.mcs.anl.gov/~pieper/df.html>
- [22] Jazayeri, M., and Lugmayr, W., *Gypsy: A Component-based Mobile Agent System*, 8th EuroMicro Workshop on Parallel and Distributed Processing (PDP2000) (Rhodos, Greece, January 19 -21, 2000).
<http://www.infosys.tuwien.ac.at/Gypsy/docu.html>
- [23] Karnik, N., and Tripathi, A., *Design Issues in Mobile-Agent Programming Systems*, IEEE Concurrency, July 1998.
- [24] Korpela E., Werthimer D., Anderson D., Conn J., and Lebofsky M., *SETI@home: Massively Distributed Computing for SETI*, In IEEE Computing in Science and Engineering, Vol3, No. 1, 30 September 2001,
<http://www.computer.org/cise/seti.htm>.
- [25] Leopold, C., *Parallel and Distributed Computing, A survey of Models, Paradigms, and Approaches*, page 127-143, 2001.

- [26] Mishra, S. and Xie, P., *Interagent Communication and Synchronization in DaAGent*, Department of Computer Science, University of Colorado, Boulder, CO, USA.
- [27] MOSIX R&D Group, What is MOSIX? 2002, <http://www.cs.huji.ac.il/mosix>.
- [28] Ruan, J., M.S., *Spider III: A Multi-Agent-Based Distributed Computing System*, Master Project. Dept of Computer Science, California State University, San Bernardino, June 2002.
- [29] Sarmenta, L., Bayanihan: *Web-based Volunteer Computing*, Ph.D. Dissertation. Dept. of Electrical Engineering and Computer Science, MIT, March 2001.
- [30] SETI@home, *SETI@home: Search for Extraterrestrial Intelligence*, <http://seti@home.ssl.berkeley.edu>.
- [31] Silva, L., Soares, G., Martins, P., Bautista, V., and Santos, L., *Comparing the Performance of Mobile Agent Systems: A Study of Benchmarking*, Departamento Engenharia Informatica Universidade de Coimbra - POLO II Portugal
- [32] Thearling, K., Ph.D., *An Introduction to Data Mining*, <http://www.thearling.com/dmintro/index frame.htm>

- [33] Toshiba Bee-gent
<http://www2.Toshiba.co.jp/plangent/index.htm/>
- [34] Toshiba Plangent
[http://www2.Toshiba.co.jp/plangent/index.htm.](http://www2.Toshiba.co.jp/plangent/index.htm)
- [35] Wang, K., M.S., *SPIDER II*, Master Project,
California State University, San Bernardino, CA,
2001, <http://web.csusb.edu/pool/grad/kwang>.
- [36] Woods, A.W., *What is in a Link: Foundation fro*
Semantic Networks, in *Representation and*
Understanding, Bobrow & Collins (Eds), Academic
Press, 1975.
- [37] Yu, H., *Spider: An Overview of an Object-Oriented*
Distributed Computing System, Master Thesis,
Department of Computer Science, California State
University, San Bernardino, CA, 1997.
- [38] Zhang, W.R., *A Cognitive-Map-Based Approach to the*
Coordination of Distributed Cooperative Agents,
IEEE Trans. Systems, Man and Cybernatics, Vol. 22,
No. 1, Jan/Feb 1992.

DEVELOPMENT OF EDUCATIONAL CURRICULUM AND STANDARDS
OF PRACTICE FOR THE MANAGEMENT OF ACUTE CONFUSION
SYNDROME/DELIRIUM AMONG HOSPITALIZED PATIENTS

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Nursing

by
Nora Nurten Moti
March 2003

DEVELOPMENT OF EDUCATIONAL CURRICULUM AND STANDARDS
OF PRACTICE FOR THE MANAGEMENT OF ACUTE CONFUSION
SYNDROME/DELIRIUM AMONG HOSPITALIZED PATIENTS

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Nora Nurten Moti

March 2003

Approved by:

Susan Lloyd, PhD, RN, CNS
Susan L. Lloyd, PhD, RN, CNS, Chair,
Nursing

3-6-03
Date

Marcia Raines PhD RN
Marcia Raines, PhD, RN

Shirley Bristol JD, RN, MSN, CNS
Shirley Bristol, MSN, RN, JD, CS

ABSTRACT

The primary objective of this project is the development and implementation of an educational program for the staff nurses for the effective management of Acute Confusion (AC)/Delirium among hospitalized patients at Kaiser Hospital in Fontana.

A secondary objective is to provide nursing staff with a screening tool to assess AC/Delirium and follow the standards of practice protocol to minimize complications, i.e. falls, restraint and sitter use, thus, improve clinical outcomes. The development of these materials included a systemic review of management of Acute Confusion, delirium, agitation and restraint reduction articles in electronic databases (Medline, CINAHL, etc.). Full text reviews were done on 48 studies. Educational curriculum was developed with cooperation of the Nursing Managers, Hospital and Nursing Quality Improvement Managers, Educators, Clinical Nurse Specialist and Union Partners at Kaiser Permanente in Fontana. This protocol will be used to help nursing staff implement the best evidence based practice available in the care of confused patients. This educational program and standards of practice have been accepted by the Kaiser Permanente,

Fontana Education and Training Department for immediate implementation.

ACKNOWLEDGMENTS

The help of the following people is deeply appreciated. My husband for his relentless encouragement, Dr. S. L. Lloyd for her guidance in completing this project and Ms. N. Gonzalez for her patience and care in the typing.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	v
CHAPTER ONE: BACKGROUND	
Introduction	1
Purpose of the Project	3
Scope of the Project	4
Significance of the Project	5
Limitations of the Project	5
Definition of Terms	6
CHAPTER TWO: REVIEW OF THE LITERATURE	
Introduction	8
Incidence and Risk Factors of Acute Confusion/Delirium	8
Tools to Measure Acute Confusion/Delirium	9
Nurses Role in Managing the Effected Population	11
The Context of the Education Programs	12
Social Learning Theory	14
Summary	15
CHAPTER THREE: METHODOLOGY	
Planning of the Project	16
Development of the Project	17
Management of the Educational Course	19
Implementation of the Project	20
Summary	22

CHAPTER FOUR: RESULTS AND DISCUSSION	
Introduction	23
Presentation of the Findings	23
Discussion of the Findings	23
Summary	24
CHAPTER FIVE: EVALUATION, CONCLUSIONS AND RECOMMENDATIONS	
Introduction	25
Evaluation	25
Summary, Conclusions and Recommendations	26
Summary	26
Conclusions	26
Recommendations	27
APPENDIX A: AGENCY PERMISSION FORM	29
APPENDIX B: EDUCATIONAL CURRICULUM PROTOCOL FOR ACUTE CONFUSION SYNDROME/DELIRIUM	31
APPENDIX C: MENTAL ASSESSMENT TOOL MINI MENTAL STATE EXAM (MMSE)	47
APPENDIX D: ACUTE CONFUSION/DELIRIUM STANDARDS OF CARE AND PRACTICE	49
Appendix E: ACUTE CONFUSION/DELIRIUM PRE AND POST TEST	52
APPENDIX F: PROJECT REFERENCES	57
REFERENCES	60

CHAPTER ONE

BACKGROUND

Introduction

Acute Confusion (AC), also known as delirium, is a multifaceted, commonly occurring phenomenon and a major contributor to poor healthcare outcomes among hospitalized geriatric patients (Ludwick, 1999; Foreman et al., 1999). It is estimated that depending on the population studied, 24-80% of all elderly hospitalized clients experience some degree of confusion (Evans et al., 1993). Indeed, approximately 50% of all patients admitted with hip fractures develop confusion during the course of their hospitalization (Rateau, 2000).

AC/Delirium has long been associated with poor patient outcomes including increased length of stay, morbidity and mortality. The management of AC/Delirium is becoming a major area of concern of healthcare professionals (Justic, 2000).

Yet, AC/Delirium is frequently under diagnosed and inappropriately treated resulting in increased medical complications, poor patient outcomes, higher nursing care hours and greater financial costs. One study showed that hospitals could suffer \$30,000 loss for every confused

patient. Thus, one extra day of length of stay (LOS) for each confused patient can cost Medicare up to \$2 billion annually (Foreman & Zane, 1996).

Confused patients more frequently develop pressure ulcer infections, experience adverse reactions to therapeutic doses of medications and falls. Because they are not able to think clearly they exhibit unsafe behaviors such as, pulling intravenous lines and other tubes and wandering out of their rooms. They require close nursing observation such as sitters or they are physically restrained to prevent them from harming themselves (Sullivan-Marx, 1994).

It is also estimated that one in five elderly clients are physically restrained at some time during their hospitalization due to aggression, confusion, agitation and withdrawal (Mion & Strumpt, 1994; Strumpt & Evans, 1988).

Reducing the use of physical restraints has become a major focus of accrediting and regulatory agencies, i.e., Joint Commission on Accreditation of Healthcare Organizations (JCAHO). The study showed that in two acute care hospitals, there was an approximate 20% reduction in restraint use at the end of the implementation of a comprehensive program. The program included a

confusion/delirium assessment, management strategies and fall prevention strategies in addition to anchoring/securing therapeutic devices, i.e., peripheral intravenous (IV) lines, nasogastric tubes and indwelling bladder catheters (Mion et al., 2001).

Purpose of the Project

The purpose of this project was to develop an educational curriculum and standards of practice for a multidisciplinary, comprehensive approach to managing Acute Confusion/Delirium among hospitalized patients for implementation among the nursing staff at the Kaiser Permanente Fontana Medical Center.

Previously collected data assessing staff nurses knowledge level of Acute Confusion (AC) including its etiology, assessment and differentiation from dementia, demonstrated the need for a comprehensive educational program. This program requires participation of other disciplines and support of all the stakeholders, i.e., Physicians, Staff, Managers, Educators, the Clinical Nurse Specialist and Quality Improvement professionals.

At the Kaiser Permanente Fontana facility, the number of acute inpatient falls in the first quarter of 2001 increased by two fold comparing to the same period a year

ago (1st Quarter QI Report, 2001). In addition, in 2001, approximately \$1.5 million was spent on sitters for confused patients who required close monitoring. At present at the Fontana Medical Center, there are fragmented nursing management strategies dealing with fall prevention, reduction of physical restraints and sitter use in addition to monitoring of wandering and missing patients. All these activities hold the common thread of "confusion."

Scope of the Project

The scope of this project includes:

1. A review of previously collected data related to Critical Care staff nurses knowledge level of Acute Confusion/Delirium among hospitalized patients.
2. Development of an educational curriculum program for Critical Care staff nurses related to Acute Confusion/Delirium among hospitalized patients.
3. Development of specific standards of practice for Critical Care staff nurses to use when hospitalized patients demonstrate Acute Confusion/Delirium at Kaiser Permanente,

Fontana, based on previously published standard tools.

4. Development of a pre-post test for Critical Care staff nurses to evaluate personal knowledge of Acute Confusion/Delirium before and after completion of the educational curriculum.

Significance of the Project

Most nurses function as a generalist (Rapp et al., 2001). The Advance Practice Nurse (APN) has been conceptualized within five complementary role components: expert clinical practice, education, research, consultation and clinical leadership (CA BRN, 1998). Therefore, one of the primary activities of the APN is to provide expert care, education consultation and leadership in program development. APN's act as consultants in developing multidisciplinary comprehensive approaches to management of confused hospitalized patients.

The implementation of this program hopefully will have an impact on positive patient outcomes.

Limitations of the Project

Some project limitations included the size and the decision making structure of the organization's guidelines in developing and implementing projects and the multi

level (local, divisional and regional) oversight committees involvement in approving the project for implementation which caused a tremendous amount of time delay in developing this project. Furthermore, overall assessment data had weak external validity due to being conducted in Critical Care units, and then generalized to the entire medical-surgical inpatient areas.

Definition of Terms

The following terms are defined as they apply to the project.

Acute Confusion (AC)/Delirium is defined as a reversible confusional state, which is usually noted between the third and seventh day after admission to the hospital and generally resolves within 48 hours after discharge (Ballard, 1981).

The term acute confusion state, acute confusion, and delirium are used interchangeably by most healthcare professionals in describing a specific type of cognitive impairment. One of the easiest ways to characterize the difference is; nurses tend to view this cognitive phenomenon more broadly and call it acute confusion whereas physicians view the phenomenon as delirium (Rapp et al., 2001). However,

it is important to differentiate Acute Confusion/Delirium from dementia.

Acute Confusion/Delirium is a condition of cerebral insufficiency, characterized by the basic properties of impairment of cognitive process and it is potentially reversible.

Dementia is the loss of intellectual functions resulting from gross change in the physical functioning of the brain. It is organic and permanent whereas AC/Delirium is situational and transient.

Disorientation to time, place and person has been regarded as the primary distinguishing feature of AC/Delirium. A patient with delirium is usually more restless than one who is with dementia (Nadelsen, 1976).

AC/Delirium develops in 32% to 50% of patients who have dementia. AC/Delirium that is superimposed on preexisting dementia is difficult to recognize and differentiate from a patient's baseline dementia (Justic, 2000). Therefore, it is necessary to perform a baseline mental assessment on patients.

CHAPTER TWO

REVIEW OF THE LITERATURE

Introduction

The review of the literature related to Acute Confusion/Delirium will focus on: a) the incidence and risk factors, b) presently available tools to measure mental status, c) nurses role in managing the effected population theoretical framework, d) the context of the educational program, and e) social learning theory related to Critical Care nurses.

Incidence and Risk Factors of Acute Confusion/Delirium

It is estimated that 24% to 80% of hospitalized patients experience some degree of AC (Pomzei et al., 1994; Csokasy, 1999). Hospitalized patients are at very high risk for the development of AC/Delirium because of factors such as multiple system illnesses, increased age, multiple drug over or under use and environmental factors. Surgical patients commonly have pain and physical discomfort. Unrelieved pain may contribute to inadequate sleep, exhaustion, agitation and confusion. Adequate pain control by use of proper analgesics is an important factor in managing AC/Delirium (Sveinsson, 1975; Blacher, 1980).

In Critical Care, mechanically ventilated patients require continuous assessment and interventions in managing their pain and discomfort. Although opioids may produce sedating effects, they do not provide amnesia. Therefore, it is recommended that sedation of agitated critically ill patients should be started only after providing adequate analgesia and treating reversible physiological causes (Jacobi et al., 2002).

Risk factors of AC are (a) increasing age, (b) increasing severity of illness, (c) multiple chronic illness, (d) history of cognitive impairment (dementia and depression) or previous experience of confused patients, (e) substance and alcohol use (Foreman et al., 1999). Primary factors could be grouped under four major categories: (1) physiological, (2) psychological, (3) pharmacological, and (4) environmental factors (Rogers & Bocchine, 1999).

Tools to Measure Acute Confusion/Delirium

Three instruments which have acceptable psychometric properties and limited user burden (Rapp et al., 2001) were reviewed. They are as follows: NEECHAM Confusion Scale which consists of three subscales that are designed to collect and evaluate the physiological, psychological

and behavioral manifestations of AC (Neelon et al., 1992). Subscale I evaluates components of cognitive status. Subscale II observes behavior and performance ability. Subscale III assesses vital functions, oxygen stability and continence. The scores may range from zero (minimal function) to 30 (normal function). Score of 0-19 points indicates AC, 20-24 mild disturbance, 25-26 points normal function (Csokasy, 1999).

Confusion Assessment Method (CAM) is a tool used for diagnosis of delirium. It has four features: 1) measures acute onset of mental status changes, 2) evaluates inattention, 3) assesses disorganized thinking, and 4) evaluates the level of consciousness (Inouye et al., 1990). Administration of these tools could take up to 15 to 20 minutes (Rapp, et al., 2001).

Mini-Mental State Examination (MMSE) is a general mental status screen. It evaluates the following: 1) orientation, 2) registration, 3) attention and calculation, 4) recall, and 5) language skills. This 30-point mental state exam used as a practical method for grading the cognitive state of patients for the clinicians. Concurrent validity was determined by correlating MMSE scores with the Wechsler Adult Intelligence Scale of verbal and performance scores. MMSE

versus verbal IQ Pearson r was 0.776. And MMSE versus performance IQ Pearson r was 0.660. The MMSE reliability for multiple tests by multiple examiners had a Pearson r correlation coefficient of 0.887 (Folstein, Folstein, & McHugh, 1975).

MMSE to assess baseline cognitive functioning of patients could be administered in 5-10 minutes (Barker, 1994).

Nurses Role in Managing the Effected Population

In spite of the large number of confused patients in acute care hospitals much remains unknown about the nurses ability to recognize and treat confusion (Ludwick & O'Toole, 1996). Nurses who skillfully assess the mental status of their patient's play an important role in changing the way AC is diagnosed and treated (Levkoff et al., 1992).

In the Linkage Model (Crane, 1985) performance improvement through clinical research utilization, the Advance Practice Nurse (APN) acts as a consultant who links the resource system, research studies, findings and recommendations with the user system, clinical practice (Jones, 2000).

Critical Care nurses working with confused patients often times do not assess their patient's for Acute Confusion, thus inappropriate nursing interventions, i.e., application of chemical or physical restraints are commonly used.

The Context of the Education Programs

Several areas of investigation have been pursued in the area of assessment and proper management of AC/Delirium. Vermersch and Henley (1997) validated the clinical assessment of confusion - A (CAC-A) an observational checklist developed for nurses to measure presence and level of confusion in hospitalized adults. They concluded that AC is a multidirectional phenomenon reflecting at least cognition, motor activity, general behavior, and orientation. Recognition and treatment of the phenomenon not only lives in the truth of the confusion itself, but also in the nurse's view of it. Mion and her colleagues concluded that group educational activities conducted for medical and nursing staff, focusing on AC/Delirium, agitation and falls, the risk factors, proper strategies used to guide nurses' practice were effective in bringing about a 20%-59% reduction in

physical restraint use on the medical floor in two acute care hospitals (Mion et al., 2001).

Csokasy (1999) in response to the inconsistent identification and monitoring of AC, conducted a study in which the NEECHAM Confusion Scale was used to assess the cognitive status of geriatric hospitalized patients. The results of this study indicated that 47% of elderly ICU patients became confused most often within 24 hours of admission. She hypothesized that as nurses become more knowledgeable about AC, they will develop better assessment skills in behavioral and cognitive changes of their patients indicating AC.

Foreman et al., (1999) published their article entitled "Standard of Practice Protocol: Acute Confusion/Delirium" which provides prevention and treatment guidelines of AC patients in order to improve nursing care for this population. They provide guidelines for prevention, elimination and minimization of the etiologic agents, including administering medications judiciously, preventing infections, and maintaining fluid and electrolyte balance in addition to providing therapeutic environment and supportive nursing care. The expected outcomes are improved patient care, increased detection of AC/Delirium, implementation of appropriate

interventions, decreased length of stay, cost, morbidity and mortality.

Social Learning Theory

Social Learning Theory emphasizes that individual's beliefs about their capabilities or their confidence are the best predictors of their actual behavior (Bandura, 1977). Self Efficacy Theory, which is born out of the principles of the Social Learning Theory also emphasizes the individual's beliefs about their personal capabilities can forecast future behavior (Bandura, 1977). Thus increased knowledge of nurses in assessing AC, it's clinical presentation, disease course and progression, appropriate nursing interventions would predict positive outcomes (Rapp et al., 2001).

In Social Learning Theory much behavior is motivated and regulated by internal standards and self-evaluative reactions to the individual's own actions, including self incentives and self-concepts of efficacy (Redman, 1993). Critical Care nurses require skills and perceived self efficacy when delivering competent care to the confused patients. Perceived self efficacy is a belief that Critical Care nurse has the ability to realize the optimal

level of competency and performance in managing confused Critical Care patients.

Summary

Hospitalized patients are at a high risk for development of AC/Delirium due to various factors. These risk factors increase by increased age and severity of their illness. Critical Care patients are among the high risk patients in developing AC/Delirium. Therefore, Critical Care nurses need to be educated to assess the risk factors by utilizing tools available to them. Nurses who are skillful in assessing the mental status of their patients and utilize the knowledge and skills attained by this educational program will be more competent in delivering high quality nursing care to their confused patients, therefore, achieve better patient outcomes.

CHAPTER THREE

METHODOLOGY

Chapter Three documents the steps used in developing this project specifically, planning, development, and management and implementation of the educational course.

Planning of the Project

Permission to do this project was obtained from the Hospital Administration Risk Management Director at Kaiser Permanente, Fontana Hospital (see Appendix A). The planning and development of the project required solicitation of input from all the stakeholders. Department Administrators, Nurse Educators, Clinical Nurse Specialist, Quality Improvement Coordinator, Co-chair Union Partner Staff Nurse and the Risk Management Director, were consulted for their initial and continuous input into the project.

A transition management plan starts where people are and then works forward, step by step, through the process of leaving the past behind, getting through the restraint period, profiting from it and emerging with new attitudes, behaviors and identity. A transition management plan can be initiated from the current possibilities (Bridges, 1994).

Previously collected data assessing staff nurses knowledge of recognition, epidemiological aspects and effective nursing intervention in managing AC/Delirium indicated that there was a need for development of an educational curriculum at Kaiser Hospital in Fontana. Even though there were Patient Falls Protocol, Physical Restraints Protocol, Sitters Protocol, there was not a Standard of Practice Protocol of Acute Confusion/Delirium phenomenon. Where AC/Delirium was the main cause of falls, use of restraints and sitters.

The initial stage of this project required a meeting with the Hospital Quality Improvement Chair who is an MD and the Nursing Quality Improvement (QI) Director who is an RN to discuss the need, purpose, goals and objectives and obtain their support. Subsequent meetings were held with the Risk Management Coordinator, Nursing Director and the Nursing QI Co-chair who also is a union partner. Due to the Union-Management Partnership Agreement, no project could be developed without the union leadership's participation at Kaiser facilities.

Development of the Project

The project was born in April of 2001 as a result of the needs assessment survey in Critical Care areas

identifying the necessity of development of an educational program to train staff nurses to improve the clinical outcomes of their confused patients.

Problem identification began by analyzing the previously collected data in Critical Care and it progressed to the development of pre-test, post-test assessment of the pre-educational curriculum knowledge and level of confidence of Critical Care nurses in managing confused patients. The second step was to facilitate the communication link between the resource system and the users. The resource system consisted of web sites, which supplied access to relevant literature, research, evidence-based practice, and case studies. Users were identified as all the stakeholders of the project. It consisted of the Critical Care Manager, Clinical Nurse Specialist (CNS), Educator, Staff Nurses, Respiratory Therapists and the Quality Improvement Coordinator.

The communication link was established through committee meetings, phone calls, e-mails and one on one meetings with the previously identified parties. An educational packet was developed to address the needs of the Critical Care nurses.

Since the Advanced Practice Nurses (APN's) role often involves management and/or coordination of complex

situations (Hamric, Spross, & Hanson, 2000) it was essential to include safety measures such as prevention of falls and reduction rate of restraint and sitter use into the educational curriculum.

Management of the Educational Course

The first phase of the project concluded with the development of educational curriculum (see Appendix B). Objectives were written in Bloom's Taxonomy cognitive domain (Marzano, 2001). The curriculum includes definitions, etiology, risk factors, nursing assessment, interventions and alternatives to restraint use. A Confusion Algorithm from the literature was described. A Mental Assessment tool was provided (see Appendix C). Standards of Care and Practice were developed (see Appendix D). A pre and post-test (see Appendix E) was developed for evaluation purposes.

Learning outcomes are the result of sound teaching - learning process. And the interaction of subject matter, teaching methods and instructional material are essential for effective learning outcomes (Gronlund, 2000). The preparation stage included the linking of the course content to the organizational goal of delivering

competent, safe care to the confused hospitalized patients.

Implementation of the Project

The project had three major components:

a) Development of educational curriculum, b) Development of standards of practice, and c) Development of a pre-post test. Extensive review of the literature including Acute Confusion/Delirium National Standards developed by American Psychiatric Association in addition to Research and Evidence Based Best Practice Guidelines was conducted.

Community standards for management of confused patients were examined. An Advanced Practice Nurse expert and a Clinical Nurse Specialist of Critical Care were consulted. Nursing standards and organizational goals for patient safety for positive outcomes were incorporated into the development of the curriculum, standards of practice, and pre-post tests.

Development of Standards of Practice was based on the American Psychiatric Association National Guidelines for delirium (APA, 2002) and Standard of Practice Protocol for Nursing Acute Confusion/Delirium (Foreman et al., 1999).

The newly developed educational component was presented to the Nursing Leadership Team by using

Empowered Learning Model [ELM] (Covey, 1998). Colorful fliers were added to the Nurse Leadership meeting agenda. Content of the educational curriculum was presented by using colorful overhead slides produced by PowerPoint for perceptual strength (Griffin & Binder, 1998). Furthermore, questions and answer sessions provided participants the opportunity to participate fully in the learning process through sharing with others in an active dialogue.

Nursing Educators will be responsible for the teaching of the educational component of the project to the staff nurses. In addition to paid training time, two units of continuous education units (CEU's) will be offered to all nurses. The curriculum is also being incorporated into the orientation program for all new nurses, including the new graduate nurses.

Implementation will be based on Education and Training Department teaching strategies. It is expected to increase staff nurses knowledge and skills due to social teaching theory which emphasizes the individual's belief about his/her capabilities and forecast future behavior. And increased self-efficacy of the staff will translate to increased competency about the subject.

Summary

The project has potential to evolve a pilot program in Critical Care to a hospital wide program to include all confused patients. It was also expanded to include not only an educational component but also development of standards of practice and pre-post test for evaluation purposes in cooperation with the Education and Training Department.

CHAPTER FOUR
RESULTS AND DISCUSSION

Introduction

Included in Chapter Four was presentation of the result of completing the project. It would include the results of the pilot project and the expected outcomes of the planned education of the nursing staff.

Presentation of the Findings

An assessment of previously collected data at Kaiser Permanente, Fontana Hospital indicated a need for an educational curriculum for Acute Confusion/Delirium patients in a Critical Care unit. Prior to the implementation of teaching the curriculum, a pre-test will be administered by the Critical Care Nurse Educator. After the implementation of the teaching sessions which will be supplemented by guideline handouts, the same test will be given as a post-test. It is expected that there will be an improvement on the post-test which will indicate a positive impact in the knowledge level of staff nurses in Critical Care.

Discussion of the Findings

The expected improvement rate will only apply to the increased didactive knowledge of the staff nurses in

effective management of AC/Delirium in the Critical Care setting. How this knowledge will apply to the clinical practice for improved patient outcomes in all inpatient settings would be monitored by continuous data collection of prevalence rate of AC/Delirium, falls and use of restraints and sitters. The project has been accepted by the Education and Training Department for immediate implementation.

Summary

At present, it is expected that increased knowledge will translate into improved clinical practice, and that as a result of attending the classes, Critical Care nurses at the Fontana Kaiser facility will be better informed to assess, intervene and evaluate the confused and delirious patients in ICU than they were prior to attending the educational sessions of this program.

CHAPTER FIVE
EVALUATION, CONCLUSIONS AND
RECOMMENDATIONS

Introduction

Included in Chapter Five is a presentation of the evaluation and conclusions as a result of completing the project. Further, the recommendations extracted from the project are presented. Lastly, the chapter concludes with a summary.

Evaluation

Effective program evaluation has been defined as systemic investigation of the merit, worth or significance of an object. Four core standards of the program; utility feasibility, propriety and accuracy, addressed by taking six essential steps from the inception of the program (CDC, 1999).

The procedures were useful, feasible, ethical and accurate. All the stakeholders were involved from the inception of the program. In addition, need and expected effects of the project were discussed with the users. Credible evidence of Quality Improvement results and relevant literature were reviewed. New practice standards

were established with the intent to disseminate the knowledge through educational programs.

Summary, Conclusions and Recommendations

Summary

Critical Care nurses have been providing nursing care to the confused patients in Critical Care settings for a long time. When caring for a patient who is confused, it is assumed that confusion is the natural outcome of mostly elderly and frail hospitalized patients (Csokasy, 1999).

Traditional treatment consists of restraint and sitter use and psychotropic medication administration. The cognitive status assessment, and comprehensive nursing standards of practice protocol have been nonexistent in Kaiser Hospital Critical Care settings.

Therefore, this project was developed to provide nurses with an educational curriculum, standards of practice protocol and an evaluation tool, pre-post test to measure their cognitive progress in effective management of AC/Delirium for positive patient outcomes.

Conclusions

The conclusions extracted from the project are as follows:

- 1) The educational curriculum and standards of practice as developed for Critical Care nurses have been accepted for implementation.
- 2) The project in its entirety is to be used by the Quality Improvement and Risk Management Department Directors to be shared in the upcoming JCAHO inspection as proof of a Quality Improvement project in response to needs assessment of clinical practice.
- 3) The project was registered with the Service Quality Department Advisor to be added to the database to be eligible for the Excellence Award.

Recommendations

Recommendations resulting from the project are as follows:

- 1) There was an agreement by the Nurse Executive and Nurse Educators that the course will be extended to four hour educational sessions with the full support of the Education and Training Director, starting January 2003. Due to budgetary restraints in late 2002, the course could not be made mandatory.

- 2) It is hoped that Quality Improvement indicators of falls, restraints and sitter use will be collected to compare results of previous assessment data to post project implementation data for expected quantitative improvement scores.

APPENDIX A
AGENCY PERMISSION FORM

Kaiser Foundation Hospitals
9961 Sierra Avenue
Fontana, California 92335



Dr Marcia Raines
California State University
San Bernardino
5500 University Parkway
San Bernardino, California

June 25, 2002

Dear Dr. Raines;

Nora Moti, RN has been working on a thoroughly comprehensive project for the "Management of Confused Patients" with the permission of Hospital Administration and Risk Management.

If you have any questions regarding this matter, please feel free to contact me at 427-7809.

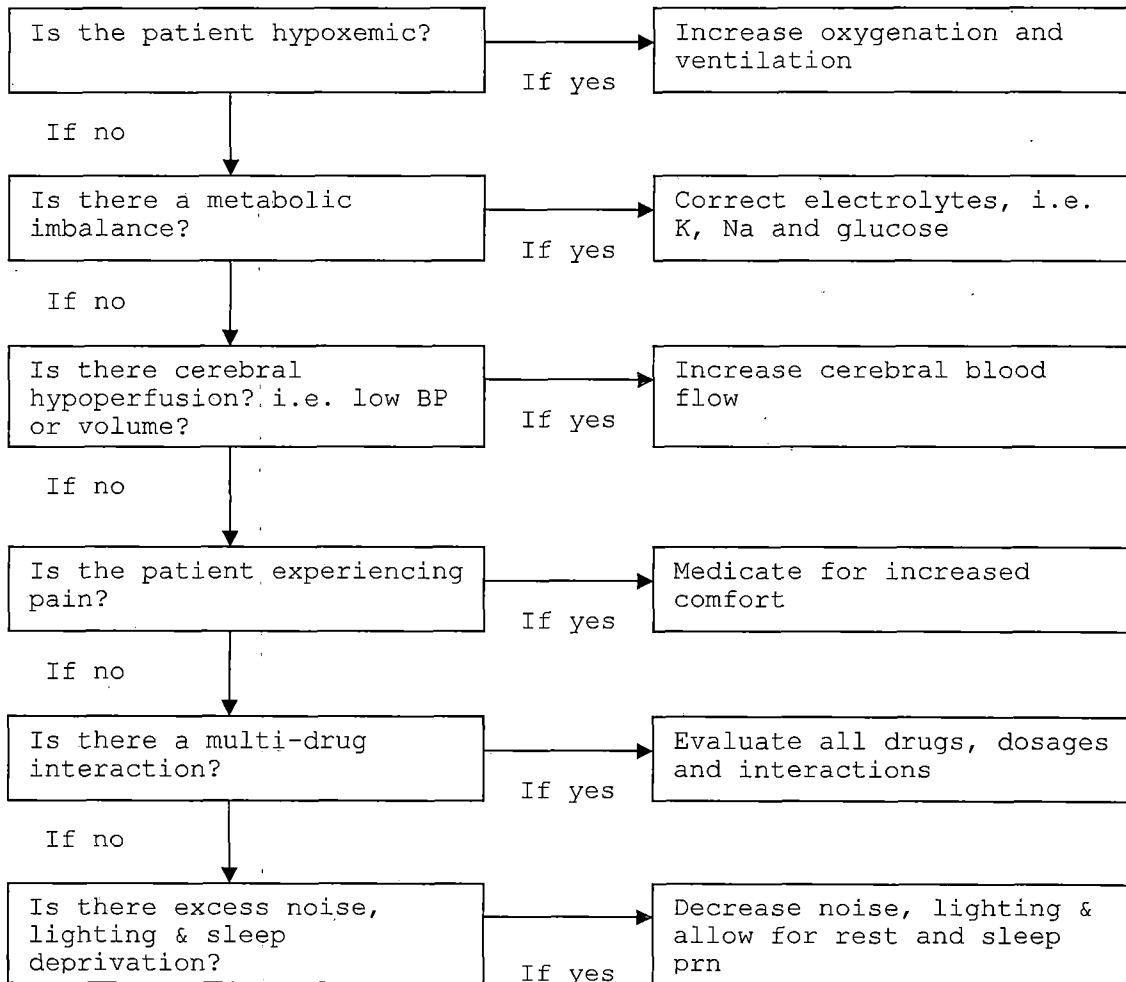
Thank you,

Kim Sheets RN, MBA

Kim Sheets RN, MBA
Director of Risk Management
Hospital Administration

APPENDIX B
EDUCATIONAL CURRICULUM
PROTOCOL FOR ACUTE CONFUSION
SYNDROME/DELIRIUM

**POSSIBLE CAUSE FACTORS OF
ACUTE CONFUSION/DELIRIUM**



Sources: 1. APA National Guidelines
 2. Foreman, M.D., Mion, L.C., Tryestad, L., Flecher, K. (1999). *Standard of Practice Protocol: Acute Confusion/Delirium. Geriatric Nursing, 20 (3) 147-152.*

Adapted by N. Moti (June, 2002)

Educational Curriculum and Standard of Practice Protocol for Acute Confusion Syndrome/Delirium

*Goal - Comprehensive Multidisciplinary Curriculum Development
for Nursing Management of Acute Confusion/Delirium*

Objectives

- I) Define acute confusion/delirium versus dementia
- II) Identify etiology and primary risk factors contributing to acute confusion/delirium
 - 2.1 Identify physiological factors
 - 2.2 Identify psychological factors
 - 2.3 Identify pharmacological factors
 - 2.4 Identify environmental factors
- III) Describe nursing process in management of Acute Confusion/Delirium
 - 3.1 Describe assessment/identification of acute confusion, delirium
 - 3.2 Mental Assessment Tool (see Appendix C)
 - 3.3 Describe appropriate nursing intervention
 - 3.4 Describe evaluation of interventions

Definitions

Acute Confusion (AC)/Delirium is a condition of cerebral insufficiency, characterized by the basic properties of impairment of cognitive process, potentially preventable and reversible, confusional state which is normally noted after 24-72 hours following admission and resolves 48 hours after discharge (Ballard, 1981).

Dementia is the loss of intellectual functions resulting from gross changes in the physical functioning of the brain. It is organic and permanent (Nadelsen, 1976).

Prevalence Rate: It is estimated that depending on the population studied 24-80% of all elderly patients' experience some degree of confusion (Evans et al. 1993). 50% of all patients admitted with hip fractures develop confusion during the course of their hospitalization (Pateau, 2000). About 50% of the patients with dementia will develop AC/Delirium (Justic, 2000)

Poor Outcomes: AC/Delirium has long been associated with poor patient outcomes including increase in falls, sitter use, length of stay (LOS), morbidity and mortality. It is estimated that one in five elderly patients is physically restrained during their hospitalization due to confusion, agitation and withdrawal (Mion & Strumpt, 1996).

Etiology

The cause of acute confusion/delirium is often difficult to determine. The elderly seem to be especially prone for many reasons such as multiple chronic medical problems, polypharmacy, altered metabolism, poor nutrition, and increased vulnerability to stressors. Acute physiologic problems, stress or changes in routine may trigger an episode of acute delirium in this population. Patient may experience illusions, hallucinations and delusions (Foreman, 1993). Nocturnal AC/Delirium, also known as "Sundown Syndrome" where confusion is seen primarily and exclusively after dark (Beek-Bates & Rogers, 1990).

Risk Factors

AC/Delirium usually results from the interaction of physiological, psychological and socioenvironmental factors, not just one factor. The clinical significance of this is that removing or treating one factor in isolation may not be sufficient to resolve the AC/Delirium. The four major categories of risk factors are as follows:

- 1) Physiological factors (Foreman, 1999; Marcantonio, E.R., et al, 1996)
 - a) Decreased O₂ (hypoxia, peripheral or central cyanosis, accessory muscle use for breathing, tachypnea, decreased PO₂, paradoxical breathing)
 - b) CV problems (CHF, dysrhythmias, cardiac surgery, hypo or hypertension, hypo perfusion to brain)
 - c) Pulmonary problems (pneumonia causing impaired gases exchange thus decrease oxygenation of vital organs including brain)
 - d) Metabolic imbalances (acidosis, alkalosis, renal or liver failure, electrolyte imbalance, hypokalemia K⁺<3.5 mEQ/L., hypernatremia Na⁺>146mEQ/L., nausea, vomiting, anorexia)
 - e) Infection (most commonly urinary tract and respiratory frequency, urgency, nocturia, incontinence)
 - f) Drug intoxication's or withdrawal (ETOH and drugs, delirium, tremors, marked agitation)
 - g) GI bleed (decreased blood volume, low cerebral perfusion, dehydration)
 - h) Endocrine disturbances (diabetes, hypo-hyper glycemia, hypo or hyper thyroidism)
 - i) Nutritional deficiencies (lack of nutrients, protein, vitamins and minerals for proper brain function)

- j) Altered temperature regulation (possible cerebral involvement in regulating body temperature or metabolic disturbances)
 - k) Ineffective processes and physiologic stress (improper comfort measures)
 - l) Pain (unmanaged or poorly managed pain control)
- 2) Psychological factors (Fisher & Flowerlew, 1995)
- a) Grief, fatigue (due to loss of health and comfort)
 - b) Anxiety (due to unknown progression of health problems, procedures)
 - c) Voicelessness/fear (loss of independence and control)
 - d) Addiction to drugs/alcohol (withdrawal)
 - e) Personal problems - insecure family/work (depending on the severity of illness, fear of losing work or family support)
 - f) Financial situations (possible loss of job and income)
 - g) Depression, paranoia (psychological problems due to severe illness, trauma or surgery)
 - h) Aggressive, dominant personality (independence vs dependence when in the hospital)
- 3) Physical environment (Rapp, et al, 2001)
- a) Sleep deprivation (hospital procedures, loud noises causing lack of sleep and rest periods)
 - b) Increased noise (nurses and other healthcare workers, mechanical noises, i.e. ventilators, IV pumps)
 - c) Constant lighting (lights on 24 hours a day)
 - d) Unfamiliar environment (different environment than the one patient used to)
 - e) Sensory deprivation or overload (misperception of visual and auditory stimuli, loud noises)
 - f) Communication impairment (endotracheal tubes unabling patients to communicate)
 - g) Immobilization, dependency (altered mobility, restraints)
 - h) Physical presence of tubes, etc.

- 4) Pharmacological factors (Rapp, et al, 2001; Kane, A.M, Kurlowicz, L.H 1996)
- a) Pain medications (Demerol, Morphine causing altered mental status, confusion, paranoia)
 - b) Decadron psychosis, euphoria, insomnia, mood swings, personality changes, severe depression to frank psychotic manifestation.
 - c) Adding more than three medications during hospital stay (drug intoxication).
 - d) Anticholinergic drugs (atropine, scopolamine, and related compounds), antitussives, anti-Parkinson agents, drugs to control cardiac output and rhythm, the tricyclic antidepressants, and antihistamines may not be recognized as potential causes of untoward psychological effects.

Fisher, B.W., & Flowerdew, G. (1995). A simple model for predicting post operative delirium in older patients undergoing elective orthopedic surgery. Journal of the American Geriatrics Society, 43(12), 175-178.

Foreman, M.D., Mion, L.C., Tryestad, L., Flecher, K. (1999). Standard of practice protocol: Acute confusion/delirium. Geriatric Nursing, 20(3), 147-152.

Kane, A. M., & Kurlowicz, L.H. (1994). Improving the postoperative care of the acutely confused older adults. MEDSURG Nursing, 3(6), 453-458.

Mancantonio, E.R., Goldman, K., Mangione, C.M., Ludwig, L.E., Muraca, B., Haslwuer, C.M., Donaldson, M.D., Whittlemore, A.D., Sugarbaker, D.J., Poss, R., Hass, S., Cook, E.F., Orau, E.HJ., & Lea, T.H. (1996). A clinical prediction rule for delirium after elective non cardiac surgery. Journal of the American Medical Association, 271(2), 136-139.

Rapp. C.G., Onega, L., Trip-Reiner, T., Mobily, P., Wakefield, B., Kundsat, M., Akins, J., Wadle, K., Mentes, J.C., Culp, K., Meyers, J., & Waterman, J. (2001). Training of acute confusion resource nurses. Knowledge, perceived confidence and role. Journal of Gerontological Nursing, (April) 24-40.

Potential Psychological Side Effects of Pharmacological Agents Commonly Used

Name	Effect
Aminocaproic acid	Acute confusion/delirium, hallucinations
Anticonvulsants	Tactile, auditory, and visual hallucinations, delirium, agitation, paranoia, confusion
Antihistamines	Anxiety, hallucinations, delirium
Atropine and anticholinergics	Confusion, memory loss, delirium, auditory and visual hallucinations, paranoia
Barbiturates	Visual hallucinations, depression
Cephalosporins	Confusion, disorientation, paranoia
Corticosteroids	Depression, confusion, paranoia, hallucinations
Digitalis glycosides	Nightmares, confusion, paranoia, hallucinations
Diazepam	Hallucinations, depression
Lidocaine	Disorientation, hallucinations, paranoia
Methyldopa	Hallucinations, paranoia, nightmares
Morphine	Transient hallucinations, disorientation, visual disturbances
Penicillin G	Hallucinations, disorientation, confusion, agitation
Albuterol	Hallucinations, paranoia
Demerol	Acute confusion, delirium

Sources: Easton C. & Mackenzie (1998). Sensory perceptual alterations. Delirium in the Intensive Care Unit. Heart and Lung, 17(3), 229-237.

Adapted by N. Moti (June, 2002)

Distinguishing Between AC/Delirium and Dementia

Characteristics	AC/Delirium	Dementia
Onset	Acute, develops quickly	Insidious
Duration	Hours to days, usually less than one week	Ongoing, irreversible
Course	Worse at night	Essentially stable; may be progressive over time
Associated conditions	Systemic illness commonly present	No systemic condition necessary
Attention span	Attention deficit; unable to focus or shift attention, easily distracted	Typically unaffected until late stages
Arousal level	Fluctuates from lethargy to agitation	Normal until late stages
Orientation to person, place and time	Variably impaired for person and place, almost always for time	Variably impaired for person and place, almost always for time
Cognition	Disorganized thoughts; hallucinations and delusions common	Usually ordered; content impoverished, abstraction poor; loss of common knowledge; may confabulate
Speech and language	Dysarthric, disorganized, slow or rapid, often inappropriate and incoherent	Ordered except in advanced dementia; may have word finding difficulty
Memory	Temporarily impaired	Loss of recent memory; remote memory impaired in later stages
Treatment	Protect the patient and treat the causes	Protect the patient and treat the behaviors

Sources: Foreman, M.D. & Zane, D. (1996). Nursing strategies for acute confusion in elders. American Journal of Nursing 96 (4), 44-51.

Adapted by N. Moti (June, 2002)

Nursing Process for Managing AC/Delirium

Treatment needs to encompass two aspects concurrently; identification, elimination or correction of the underlying cause(s) and general symptomatic and supportive measures.

Assessment

- 1) Assess for the following factors, which would place patients at high risk for acute confusion/delirium.
 - a) Elderly persons (60 and greater) especially post-hip fracture and post-craniotomy.
 - b) Pre-existing brain damage (especially dementia) or injury (i.e., motor vehicle injury).
 - c) Chronic alcohol and/or drug use (withdrawal symptoms can progress over a period of 24-72 hours to delirium marked by tremors acute confusional state associated with disorientation and hallucinations)
 - d) Sensory impairment (windowless ICU rooms, lack of visual stimulation, etc.)
 - e) History of acute confusion (if patient has a past history of confusion when hospitalized, it is likely to repeat)
 - f) Patient with recent hypoxic event (due to decreased oxygenation of brain cells, brain cell damage)
- 2) Assess and document the acute onset of any of the following symptoms that are a change from the patient's baseline mental status.
 - a) Difficulty paying attention/highly distractible (does the patient have difficulty focusing attention or keeping track of what is being said?)
 - b) Incoherent or rambling speech (does the patient exhibit disorganized or incoherent thinking, irrelevant conversation, illogical flow of ideas or unpredictable switching from subject to subject?)
 - c) Illusions, hallucinations (does the patient see things that are not there and imagine events that are not real)
 - 1) May be manifested by climbing out of bed, pulling at tubes, calling out for family, and refusing treatment.
 - 2) May not be obvious/overt.
 - 3) Ask patient such questions as "since your were admitted to the hospital, have you witnessed any events that seem abnormal?" or "sine your surgery, do you sometimes imagine things that you don't think can be true?" or "sometimes

when people are in the hospital they have unusual thoughts or experiences that may or may not bother them. Have you had any such experiences or thoughts?"

- 4) Disorientation (mostly time).
- 5) Episodes of agitation, especially at night.
- 6) New onset of memory problems.
- 7) Fluctuating mental state.
- 8) Appears frightened and suspicious of others.
- 9) Increasing lethargy.

(notify MD if any of these symptoms develop)

- 3) Interview consistent caregivers and family (when possible) to determine patient's baseline behavior.
- 4) In collaboration with physician, assess organic etiologic factors contributing to acute confusion by reviewing the following:
 - a) Labs, especially blood chemistries, CBC, ABG, drug levels, U/A (CBC low hemoglobin level - anemia - high WBC's - infection - ABG low O2 saturation - hypoxemia - U/A - high specific gravity - dehydration or presence of blood - system failure)
 - b) ECG, chest x-ray, CAT scan (if applicable).
 - c) Medication record (especially those with CNS/anticholinergic side effects, i.e. narcotics, cimetidine, steroids, benadryl, sleeping meds).
 - d) Vital signs, especially elevated temperature.
 - e) Pulse oximetry (note: beware of relying solely on pulse oximetry, hypercapnea can also cause acute changes in behavior and mental status and will not be reflected by the pulse ox).
 - f) Assess pain control.
- 5) Evaluate when behavior occurs, what provokes it, i.e. during ADL's, only when family present, etc.
 - a) Try to find out meaning behind behavior from patient (i.e. trying to go to the bathroom, believe you're trying to harm them, etc.).
 - b) Consider medicating patient 30 minutes prior to activity if stimulus can't be changed.
 - c) Come back at a later time.
 - d) Acknowledge patient's fear, don't dismiss it.

- 6) Assess patient's risk for fall (i.e. how stable is patient's gait? Involve PT as necessary). Institute Fall Prevention standard as necessary.
 - a) Intervention before hospital admission.
 - 1) Assess patient's mental status.
 - 2) Pre-op visit and teaching by nurses.
 - b) Interventions after hospital admission
 - 1) Increase and maintain close observation of patient until patient's confusion resolves.
 - 2) Review medication record daily with physician, discontinue, hold (as ordered by physician), or minimize use of unnecessary medication, especially those with CNS/ anticholinergic effects.
 - 3) Monitor vital signs, oxygenation, and I&O every shift and as needed. Quiet cardiac monitor alarms in the rooms.
 - 4) Don't support disorientation but don't argue with the patient (you can't talk them out of this).
 - a) Explore emotion behind patient's non-reality based statements (you may state "I can see why you say you are seeing spiders but I do not see them.")
 - b) Change the subject (re-orient to present time, place person)
 - c) Attempt to engage the patient in a diversional activity (encourage patient to comb his/her hair)
 - d) Reinforce reality-based behaviors (that they are in the hospital, their family will be here to visit them)
 - e) Enhance orientation through the use of aid such as a clock, calendar, familiar objects from home, eyeglasses, hearing aid and night light as needed.
 - f) Address patient by name (use proper name, Mr. or Mrs.)
 - g) Get patient's attention before beginning a conversation or giving direction (make eye contact)

- 5) Minimize use of restraints; use only for short-term, treatment.
 - a) Use mittens (apply mittens to hands to prevent patients from pulling lines and tubes)
 - b) Apply TAB's (security device applied to the patients that will alarm when patients attempt to leave their beds)
 - c) Consider vail bed (special bed which is enclosed at all sides with a zippered opening and a lattice tent)
 - d) Ask family to stay with patient (to observe and support patient)
- 6) Implement safety measures to protect staff and family from the patient's unpredictable behavior. Use security as needed.
 - a) Bed in low position (to prevent injury in case patient tries to get out of bed)
 - b) If you are placing side rails up routinely to keep a patient in bed, reconsider this. Patient is more likely to be injured if the side rails are up, especially if using split rails.
 - c) Provide adequate lighting, especially at night (elderly people need about 50% more lighting to see well).
 - d) Keep patient's personal items, water pitcher, call light, urinal, within reach at all times.
 - e) Make sure eyeglasses, hearing aid and other assistive devices are available and utilized.
- 7) Decrease stimulation but avoid sensory deprivation (private room if possible; provide consistent caregiver; decrease room clutter and noise level). Keep in mind that moving the patient closer to the busy nurse's station to keep a better eye on them could over stimulate them and worsen symptoms.
- 8) Normalize sleep pattern.
 - a) Recognize care, especially at night, to provide increased amounts of uninterrupted sleep (consolidate treatments, medications, vital signs).
 - b) Increase daytime activities.
- 9) Consult PT and OT (especially with elderly) as needed.
- 10) Consider psychiatric consult for psychosis, severe agitation, complex interplay or etiologic factors (for diagnostic evaluation and medication regiment) or if symptoms worsen or do not resolve within 48 hours.

- 11) Consider pharmacotherapy if the patient's physical safety is endangered. Haloperidol is considered the drug of choice for the treatment of delirium in the critically ill adult. It is particularly effective at subduing symptoms such as hallucinations, paranoid thoughts and severe agitation. In critical care areas, haloperidol should be administered through IV, as research studies indicate extra pyramidal side effects using this route are less likely.
- 12) Assess and document mental function and behavior every shift until AC/Delirium is resolved.
- 13) Provide emotional support for patients.
 - a) Take time to listen (therapeutic communication techniques help promote understanding between the nurse and patient) (Van Leuven, 2000)
 - b) Use touch
 - c) Promote physical comfort (will relieve anxiety due to pain and discomfort)
 - d) Encourage independence (will provide social wellness)
 - e) Provide opportunity for activity (will stimulate intellectual activity)
 - f) Teach relaxation techniques (the relaxation response provides calming state) (Brunner & Suddard, 1988)
- 14) "Debrief" patient/family after AC/Delirium resolves.
- 15) Involve family in decision-making, but educate family not to give detailed description of events to the patient with AC/Delirium.

Communication

- 1) Speak slowly and use simple sentences.
- 2) Always use the same word for the same object.
- 3) Give the patient time to respond. If the patient does not respond in one or two minutes, repeat the communication in the same manner.
- 4) Look directly at the patient to get his or her attention before addressing the patient.
- 5) When performing a task, use one-step commands.
- 6) Use nonverbal cues to trigger the patient's memory.

Sources: Rapp et al., 2001; Tripp-Reiner, 2001; Foreman et al., 1999

Foreman, M.D., mion, L.C., Tryestad, L., Flecher, K.
(1999). Standard of Practice Protocol: Acute
Confusion/Delirium. Geriatric Nursing, 20(3),
147-152.

Trip-Reiner, T. (2001). Acute confusion: Advancing
clinical nursing science through research
collaboration. Journal of Gerontological Nursing,
Guet Editorial, (April) 10-11.

Rapp. C.G., Onega, L., Trip-Reiner, T., Mobily, P.,
Wakefield, B., Kundsat, M., Akins, J., Wadle, K.,
Mentes, J.C., Culp. K., Meyers, J., & Waterman, J.
(2001). Training of acute confusion resource nurses.
Knowledge, perceived confidence and role. Journal
of Gerontological Nursing, (April) 24-40.

Adapted by Nora Moti (June, 2002)

ALTERNATIVES TO RESTRAINT USE

Alternatives include, but are not limited to:

- Reorientation to environment and/or plan of care (include patient in their plan of care. Orient patient to person, time and place)
- Family member present (encourage family to stay with patient)
- Scheduled ambulation, rehab or chair activity (scheduled time provides regularity in patients daily activity)
- Bowel/bladder assessment (prevents incontinence)
- Scheduled toileting (prevents patient trying to get out of bed)
- Back rub/repositioning (provides comfort and relaxation)
- Pain management (provides comfort and freedom from pain)
- Night light (provides sleep)
- Music (promotes relaxation)
- Television (provides diversion)
- Snacks (prevents hypoglycemia, provides comfort)
- Change of scenery (provides diversion, prevent boredom)
- Move patient to a room closer to the nursing desk (patient can be in a close observation)
- Use of cushions/pads (provides comfort)
- Diversional activities or reading (prevents boredom or inactivity)
- Lab values checked (monitor electrolyte imbalance)
- Familiar or needed personal object within reach (provides familiarity)
- Structured routine (prevents confusion)
- Assessment of medication side effects (to prevent adverse side effects of medications)
- Rearrangement of furniture (moving IV poles, cardiac monitors, etc. away from patient's sight reduces anxiety, confusion)
- Verbal intervention (reorientation to reality, reassurance)
- Wearing of eyeglasses, dentures, hearing aids (prevents sensory deprivation)
- Discussion with family regarding other mechanisms that were effective when utilized (Van Leuven, 2000)

Mion, L.C., & Strumpt, N. (1994). Use of physical restraints in the hospital setting: implications for the nurse. Geriatric Nursing, 15(3), 127-132.

Van Leuven, K. (2000) Clinical Companion Fundamentals of Nursing (6th ed.). Upper Saddle River, NJ: Prentice Hall, Inc.

Adapted by Nora Moti (June, 2002)

APPENDIX C
MENTAL ASSESSMENT TOOL
MINI MENTAL STATE EXAM (MMSE)

MENTAL ASSESSMENT TOOL	
Mini Mental State Exam (MMSE)	Addressograph

QUESTION	ANSWER	CORRECT YES OR NO
1. What is today's date?		<input type="checkbox"/> YES <input type="checkbox"/> NO
2. What day is it?		<input type="checkbox"/> YES <input type="checkbox"/> NO
3. Can you tell me the name of this place?		<input type="checkbox"/> YES <input type="checkbox"/> NO
4. What is your telephone number?		<input type="checkbox"/> YES <input type="checkbox"/> NO
5. What is your age?		<input type="checkbox"/> YES <input type="checkbox"/> NO
6. What is your birth date?		<input type="checkbox"/> YES <input type="checkbox"/> NO
7. Who is the current president?		<input type="checkbox"/> YES <input type="checkbox"/> NO
8. Who was the president before him?		<input type="checkbox"/> YES <input type="checkbox"/> NO
9. Spell "ROOM" backwards.		<input type="checkbox"/> YES <input type="checkbox"/> NO
10. Subtract 2 from 20 and keep subtracting 2 from each new number all the way down.		<input type="checkbox"/> YES <input type="checkbox"/> NO

Please respond to the questions below:

1. Do you consume alcohol? YES NO
a) How often? _____
b) How much? _____

2. Are you taking any sedatives or narcotics? YES NO
a) How often? _____
b) How much? _____

0-2	Errors	=	Intact
3-4	Errors	=	Mild intellectual impairment
5-7	Errors	=	Moderate intellectual impairment
8-10	Errors	=	Severe intellectual impairment

*Allow one more error if subject had no grade-school education
*Allow one fewer error if subject has had education beyond High School

MEDICAL RECORDS - PLEASE RETURN TO UNIT IF FOUND IN THE CHART

Source: Barker, E. (1994). Mental Assessment. Neuroscience Nursing. St. Louis, Mosby.
Adapted by Nora Moti (June, 2002)

Validity = 0.776
Reliability = 0.660

APPENDIX D
ACUTE CONFUSION/DELIRIUM
STANDARDS OF CARE AND PRACTICE

STANDARD: MANAGEMENT OF PATIENTS WITH MENTAL STATUS CHANGE/AC/DELIRIUM

PATIENT POPULATION: All adult patients at risk for acute confusion

<i>Standards of Care</i>	<i>Standards of Practice</i>
Any change in patient's mental status will be recognized and documented	<ul style="list-style-type: none"> • Mental status will be documented on admission by using mental status tool. • Premorbid mental status will be documented if any change occurs while hospitalized. • Description of patient behavior will be documented including: level of consciousness, attention span, ability to follow directions, presence of hallucinations, agitation.
Confused patients will be identified	<ul style="list-style-type: none"> • Purple ID band will be applied to the patient. • Patient Care Record will be tagged.
Patient will be assessed to attempt to determine cause of mental status change	<ul style="list-style-type: none"> • General assessment will include evaluation of: • Medication • O2 saturation • Pain • Infection • History of ETOH use • MD will be notified, orders will be implemented which may include lab screening (i.e. CBC, Cr, Na)
Patient will not harm himself or others	<ul style="list-style-type: none"> • Assess for the presence of behaviors which may impede caregiving: • Impulsiveness • Resistance to care • Pulling at tubes • Wandering from treatment areas - consider TABS or vail bed • /Explosive or unpredictable behavior • Provide support to prevent harm, consider: relative support to stay with patient • Patient near nurse's station for observation • Removal of any nonessential tubes

<i>Standards of Care</i>	<i>Standards of Practice</i>
	<ul style="list-style-type: none"> • Least restrictive restraint option (mittens) • Reduce demands placed on patient for nonessential procedures • Promote adequate rest • If medications are used, nursing staff will monitor target behaviors and possible side effects • Document target behavior • Titrate to minimize sedation/side effects
Patient will not feel threatened	<ul style="list-style-type: none"> • Provide the least amount of intervention that will diffuse patient's agitation • Approach patient as if he/she is frightened: approach slowly, from the front • Provide consistent caregivers • Reassure patient that this is unusual behavior and staff is attempting to find the cause
Patient's mental status will begin to return to baseline	<ul style="list-style-type: none"> • Patient will continue to be assessed re: • LOC • Attention span • Hallucinations • Ability to follow directions • Agitation • Behavior will begin to resolve and return to baseline
Patient and family will be aware of assessment of behavior, medical work up, and monitoring	<ul style="list-style-type: none"> • Education patient/family regarding symptoms and causes of AC/Delirium • Reassure family that ongoing monitoring and work up is occurring • Reassure family that acute mental status changes typically resolve with time

DOCUMENTATION: 24 Hour Patient Care Record

Source: Foreman, M.D. & Zane, D. (1996). Nursing strategies for acute confusion in elders. American Journal of Nursing 96(4), 44-51.

Adapted by Nora Moti (June, 2002)

APPENDIX E
ACUTE CONFUSION/DELIRIUM
PRE AND POST TEST

UNDERSTANDING AND MANAGING ACUTE CONFUSION/DELIRIUM TEST

Name: _____ Unit: _____ Date: _____

1. AC/Delirium is an acute cognitive impairment or state of:
 - a. dementia
 - b. confusion
 - c. pain
 - d. traumatic injury
2. Common triggers of AC/Delirium in the elderly include:
 - a. a routine without changes
 - b. pain
 - c. stress-free lifestyle changes
 - d. early postoperative mobilization
3. A patient with AC/Delirium may have both periods of hypoactivity and hyperactivity.
 - a. True
 - b. False
4. Physical restrains placed on a patient with AC/Delirium:
 - a. increase agitation and worsen delirium
 - b. are gentle to the patient's skin
 - c. should be removed at least every 4 hours
 - d. have no serious drawbacks
5. Chemical restraints such as haloperidol:
 - a. decrease sedation and confusion
 - b. have no detrimental effects for an older adult
 - c. should be administered if the patient is suffering from severe agitation
 - d. should be administered at the first sign of delirium
6. Which of the following drugs is the best choice for pain control in the elderly?
 - a. Diazepam IV
 - b. Morphine IV
 - c. Haloperidol IV
 - d. Meperidine IV
7. Which of the following interventions is appropriate when your patient exhibits signs of AC/Delirium:
 - a. check patient at least every 2 to 3 hours
 - b. move patient to a stimulating, noisy area
 - c. offer to walk patient to the bathroom every 4 to 6 hours
 - d. have patient's family or significant other stay with them

8. To help a patient with AC/Delirium you should:
- a. play loud music
 - b. keep bed in the highest position
 - c. put all the side rails up for safety
 - d. use validation and reminiscence techniques
9. To prevent your patient from pulling out the IV line, you should:
- a. restrain wrists
 - b. administer haloperidol
 - c. cover the line with gauze
 - d. use mittens or netting
10. Which of the following interventions is appropriate if a patient with delirium tries to get out of bed?
- a. restrain him/her
 - b. sedate patient with psychotropic drug
 - c. if patient is steady on his/her feet, walk with him/her
 - d. medicate him/her with meperidine
11. If you must apply a physical restraint as a last resort, JCAHO recommends using:
- a. a chemical restraint
 - b. a vest restraint
 - c. the most restrictive device possible
 - d. the least restrictive device possible
12. Which characteristic of AC/Delirium is described as dysarthric, slow and often inappropriate?
- a. memory
 - b. speech and language
 - c. cognition
 - d. attention span
13. Which of the following patients is at greatest risk for development of AC/Delirium in the hospital?
- a. a 10-year old girl who is receiving dialysis
 - b. a 50-year old man who is a victim of a multiple trauma
 - c. a 17-year old girl who is a substance abuser
 - d. a 45-year old man who has chronic cardiovascular disease
14. When attempting to determine the contributing factors related to a patient's delirium, nursing & medical staff should recognize the primary reversible factor is:
- a. acuity of illness
 - b. patient's sex
 - c. patient's medications
 - d. patient's support systems

15. Which of the following questions would most likely encourage an ICU patient to discuss his/her perception of the ICU experience?
- a. What are you thinking right now?
 - b. Do things in the ICU seem abnormal to you?
 - c. Why are you pulling at your IV tubing?
 - d. Can't you understand that no one here wants to hurt you?
16. AC/Delirium can be precipitated by sleep deprivation?
- a. True
 - b. False
17. Incidence of post-op AC/Delirium can not be reduced by preoperative orientation:
- a. True
 - b. False
18. Most important factor in preventing sensory deprivation is the total care approach of the nurse who orients the patient and explains procedures:
- a. True
 - b. False
19. AC/Delirium is twice as frequent in windowless ICU's:
- a. True
 - b. False
20. Illusions, delusions, hallucinations experienced in the ICU may not be revealed until the patient has transferred out of the ICU:
- a. True
 - b. False
21. If the cause of AC/Delirium cannot be found or treating a known cause does not help, the delirious behavior can usually be suppressed with IV Haldol and Ativan can be added as a sedative:
- a. True
 - b. False
22. Over or under medicated patients display the signs and symptoms of AC/Delirium:
- a. True
 - b. False

Foreman, M.D. & Zane, D. (1996). Nursing strategies for acute confusion in elders. American Journal of Nursing, 96 (4), 44-51.

Adapted by Nora Moti, 2002

Inpatient Nursing Quality Indicators Program
Labor-Management Program
Delirium Indicator - Answer Key to Post Test

1. b
2. b
3. a
4. a
5. c
6. b
7. d
8. d
9. b
10. c
11. d
12. b
13. c
14. c
15. b
16. a
17. b
18. a
19. a
20. a
21. a
22. a

APPENDIX F
PROJECT REFERENCES

Project References

- American Psychiatric Association. (2002). Clinical Resources: Delirium. Retrieved December 12, 2002, from http://www.psych.org/clin_res/pg_delirium_2.cfm
- Ballard, K. S. (1981). Identification of environment stressors for patients in a surgical intensive care unit. Issues of Mental Health Nursing, (3), 89-108.
- Barker, E. (1994). Mental Assessment. Neuroscience Nursing. St. Louis, MT: Mosby.
- Beel-Bates, C. A., & Rogers, A. G. (1990). An exploratory study of sundown syndrome. Journal of Neuroscience Nursing, 22(1), 51-52.
- Brunner, L. S., & Suddarth, D. S. (1988). Medical Surgical Nursing (6th ed.). Philadelphia, PA: J.B. Lippincott Co.
- Easton, C., & Mackenzie, F. (1988). Sensory perceptual alterations. Delirium in the intensive care unit. Heart & Lung, 17(3), 229-237.
- Fisher, B. W., & Flowerdew, G. (1995). A simple model for predicting post operative delirium in older patients undergoing elective orthopedic surgery. Journal of the American Geriatrics Society, 43(12), 175-178.
- Foreman, M. D. (1993). Acute confusion in the elderly. Annual Review of Nursing Research, 11, 3-30.
- Foreman, M. D. (1993). Acute confusion in the hospitalized elderly: Incidence, onset and associated factors. Research in Nursing and Health, 12, 21-29.
- Foreman, M. D., Mion, L. C., Tryestad, L., Flecher, K. (1999). Standard of Practice Protocol: Acute Confusion/Delirium. Geriatric Nursing, 20(3), 147-152.
- Foreman, M. D., & Zane, D. (1996). Nursing strategies for acute confusion in elders. American Journal of Nursing, 96(4), 44-51.
- Justic, M. (2000). Does ICU psychosis really exist? Critical Care Nurse, 20(3), 28-36.

- Kane, A. M., & Kurlowicz, L. H. (1994). Improving the postoperative care of acutely confused older adults. MEDSURG Nursing, 3(6), 453-458.
- Mancantonio, E. R., Goldman, L., Mangione, C. M., Ludwig, L. E., Muraca, B., Haslwuer, C. M., Donaldson, M. C., Whittlemore, A. D., Sugarbaker, D. J., Poss, R., Haas, S., Cook, E. F., Orau, E. J. & Lea, T. H. (1996). A clinical prediction rule for delirium after elective non cardiac surgery. Journal of the American Medical Association, 271(2), 136-139.
- Mion, L. C., & Strumtp, N. (1994). Use of physical restraints in the hospital setting: Implications for the nurse. Geriatric Nursing, 15(3), 127-132.
- Nadelson, T. (1976). The psychiatrist in the surgical intensive care. Archives of General Surgery, (111) 113-117.
- Rapp, C. G., Onega, L., Trip-Reiner, T., Mobily, P., Wakefield, B., Kundsat, M., Akins, J., Wadle, K., Menten, J. C., Culp, K., Meyer, J., & Waterman, J. (2001). Training of acute confusion resource nurses. Knowledge, perceived confidence and role. Journal of Gerontological Nursing, (April) 24-40.
- Rateau, M. R. (2000). Confusion and aggression in restrained elderly persons undergoing hip repair surgery. Applied Nursing Research, 13(1), 50-54.
- Schor, J. D., Levkofft, S. E., Lipsitz, L. A., Reiley, C., Cleary, P. D., Rowe, J. W., & Evans, D. A. (1992). Risk factors for delirium in hospitalized elderly. Journal of the American Medical Association, 267(6), 827-831.
- Trip-Reiner, T. (2001). Acute confusion: Advancing clinical nursing science through research collaboration. Journal of Gerontological Nursing, Guet Editorial, (April) 10-11.
- Van Leuven, K. (2000). Clinical Companion Fundamentals of Nursing (6th ed.). Upper Saddle River, NJ: Prentice Hall, Inc.

REFERENCES

- American Psychiatric Association. (2002). Clinical Resources: Delirium. Retrieved December 12, 2002, from http://www.psych.org/clin_res/pg_delirium_2.cfm
- Ballard, K. S. (1981). Identification of environment stressors for patients in a surgical intensive care unit. Issues of Mental Health Nursing, 3, 89-108.
- Barker, E. (1994). Mental Assessment. Neuroscience Nursing. St. Louis, MT: Mosby.
- Bastable, S. (1997). Nurse as Educator: principles of teaching and learning. Boston: Jones and Bartlett.
- Beel-Bates, C. A., & Rogers, A. G. (1990). An exploratory study of sundown syndrome. Journal of Neuroscience Nursing, 22(1), 51-52.
- Bendura, A. (1977). Self-efficacy - Toward a unifying theory of behavioral change. Psychological Review, 84(2), 191-255.
- Blacher, R. S. (1980). "Minor" psychological hazards of critical care. Critical Care Medicine, 8(6), 365-366.
- Brunner, L. S., & Suddarth, D. S. (1988). Medical Surgical Nursing (6th ed.). Philadelphia, PA: J.B. Lippincott Co.
- Centers for Disease Control and Prevention. (1999). Framework for program evaluation in public health. MMWR, 48(RR-11), 1-35.
- Covey, F. (1998). The Empowered Learning Model. A process for unbeatable performance improvement. Version (1). Franklin Covey.com Salt Lake City, UT: 1-30.
- Crane, J. (1985). Research utilization - theoretical perspectives. Western Journal of Nursing Research, 7(2), 261-268.
- Csokasy, J. (1999). Assessment of acute confusion: Use of the NEECHAM confusion scale. Applied Nursing Research, 12(1), 51-55.

- Easton, C., & Mackenzie, F. (1988). Sensory perceptual alterations. Delirium in the intensive care unit. Heart & Lung, 17(3), 229-237.
- Evans, C .A., Kenny, P. J., & Rizzulo, C. (1993). Caring for confused geriatric surgical patients. Geriatric Nursing 15, 237-241.
- Fisher, B. W., & Flowerdew, G. (1995). A simple model for predicting post operative delirium in older patients undergoing elective orthopedic surgery. Journal of the American Geriatrics Society, 43(12), 175-178.
- Folstein, M. F., Folstein, S. E., & McHugh, P. R. (1975). "Mini-mental Stage." A practical guide for grading the cognitive status of patients for the clinician. Journal of Psychiatric Research, 12, 189-198.
- Foreman, M. D. (1993). Acute confusion in the elderly. Annual Review of Nursing Research, 11, 3-30.
- Foreman, M. D. (1993). Acute confusion in the hospitalized elderly: Incidence, onset and associated factors. Research in Nursing and Health, 12, 21-29.
- Foreman, M. D., Mion, L. C., Tryestad, L., & Flecher, K. (1999). Standard of Practice Protocol: Acute Confusion/Delirium. Geriatric Nursing, 20(3) 147-152.
- Foreman, M. D., & Zane, D. (1996). Nursing strategies for acute confusion in elders. American Journal of Nursing, 96(4), 44-51.
- Griffin, D., & Binder, R. (1998). Theories of Instruction and Learning in Nursing. Los Angeles: Academic Publishing.
- Gronlund, N. E. (2000). How to write and use instructional objectives. (6th ed.). NJ: Prentice Hall.
- Hamric, A. G., Spross, J. A., & Hanson, C. M. (2000). Advanced Nursing Practice, An Integrated Approach. Philadelphia, PA: W.B. Saunders Co.

- Inouye, S. K., Van Dyck, C. H., Alessi, C. A., Balkin, S., Siegal, A., & Horowitz, R. I. (1990). Clarifying confusion: The Confusion Assessment Method. A new method for detection of delirium. Annals of Internal Medicine, 11(113), 941-948.
- Jacobi, J., Gilles, L. F., & Douglas, B. (2002). Clinical guidelines for the sustained use of sedating and analgesics in the critically ill adult. Critical Care Medicine, 30(1) 119-133.
- Jones, J. (2000). Performance Improvement through clinical research utilization: The Linkage Model. Journal of Nursing Care Quality, 15(1) 49-54.
- Justic, M. (2000). Does ICU psychosis really exist? Critical Care Nurse, 20(3), 28-36.
- Kane, A. M., & Kurlowicz, L. H. (1994). Improving the postoperative care of acutely confused older adults. MEDSURG Nursing, 3(6), 453-458.
- Levekoff, S. E., Evans, D. A., Liptzin, B., & Phillips, R. S. (1992). Delirium: The occurrence and persistence of symptoms among elderly hospitalized patients. Archives of Internal Medicine, 152, 334-340.
- Ludwick, R. (1999). Clinical decision making: Recognition and confusion and application of restraints. Orthopedic Nursing, (Jan-Feb), 65-72.
- Ludwick, R., & O'Toole, A. (1996). The confused patient: Nurses' knowledge and intervention. Journal of Gerontological Nursing, 22(1), 44-49.
- Mancantonio, E. R., Goldman, L., Mangione, C. M., Ludwig, L. E., Muraca, B., Haslwuer, C. M., Donaldson, M. D., Whittlemore, A. D., Sugarbaker, D. J, Poss, R., Haas, S., Cook, E. F., Orau, E. J., & Lea, T. H. (1996). A clinical prediction rule for delirium after elective non cardiac surgery. Journal of the American Medical Association, 271(2), 136-139.
- Marzano, R.J. (2001). Designing a New Taxonomy of Educational Objectives (2nd ed.). Thousand Oaks, CA: Corwin Press, Inc.

- Mion, L. C., Fogel, J., Sandhu, S., Palmer, R. M., Minmick, A. F., Cranston, T., Belhoux, F., Merkel, C., Berkmar, C. S., & Leipzig, R. (2001). Journal on Quality Improvement, 27(11), 605-618.
- Mion, L.C., & Strumpt, N. (1994). Use of physical restraints in the hospital setting: Implications for the nurse. Geriatric Nursing, 15(3), 127-132.
- Nadelson, T. (1976). The psychiatrist in the surgical intensive care. Archives of General Surgery, (111), 113-117.
- Van Leuven, K. (2000). Clinical Companion Fundamentals of Nursing (6th ed.). Upper Saddle River, NJ: Prentice Hall, Inc.
- Vermesch, P. E. H., & Henly, S. T. (1997). Validation of the structure for the "clinical assessment of confusion - A." Nursing Research, 46(4), 208-213.