Design and Implementation of Agent based Inter Process Synchronization Manager

Deepshikha Bhargava
Amity University Rajasthan, Jaipur
AB-531, Nirman Nagar, Kings Road,
Ajmer Road, Jaipur-302019

Madhavi Sinha, PhD
BIT-Mesra (Ranchi), Jaipur Campus
27, Malviya Industrial Area
Malviya Nagar, Jaipur

ABSTRACT
Inter Process synchronization is the coordination of simultaneous threads or processes to complete a task in order to get correct runtime order and avoid unexpected race conditions. Since processes frequently need to communicate with other processes therefore, there is a need for a well-structured communication among processes, without using interrupts [1].

This paper is an attempt to present the design and implementation of an agent based technique to provide optimized solution of Inter Process Synchronization problem, keeping the problem of critical section in to consideration.

General Terms
Inter Process Synchronization (IPS), Agents, interface agent, intelligent agent

Keywords
Inter Process Synchronization Manager (IPSM), Agent Engine, CFAE (Control Function Agent Engine).

1. INTRODUCTION
Inter Process synchronization is the coordination of simultaneous threads or processes to complete a task in order to get correct runtime order and avoid unexpected race conditions. Since processes frequently need to communicate with other processes therefore, there is a need for a well-structured communication among processes, without using interrupts [1].

The existing techniques or algorithms to solve this problem are non-agent based those have their own limitations [2]. The limitations of non-agent based solution motivated the design of an agent-based technique and the characteristics of agents compile to use an agent based technique to resolve the problem of inter process synchronization. Agent based technique [3] can add a new set of capabilities to existing applications.

Main goal of this paper is to enhance the spectrum of design tools for solving inter process synchronization problem. The Agent-based design framework described in Section 2 are constructed by combining the agent and multi-agent system concepts with particular insight from the intelligent agent based approach as design aspects of inter process synchronization problem.

Section 3 presents an implementation tool for the realization of an actual agent-based inter process synchronization manager. The conceptual framework is mapped onto an implementation framework of software components.

The application of the design framework to the design and implementation of an agent-based solution is further discussed in section 4. Further the performance evaluation of agent based IPSM is also measured on different operating system. Finally the paper is concluded and some concrete solutions are being tested and proved.

2. INTER PROCESS SYNCHRONIZATION MANAGER
IPSM stands for Inter Process Synchronization Manager which is an agent used for solving the problem of inter process synchronization. The agent based approach used in this paper is suggesting an algorithm for agent IPSM which is an attempt to propose a new approach as an optimal solution to the IPS problem. The main features of the designed system are:

- It is an agent based system. Agents are autonomous programs, they run in background and there is no need of user interface.
- Multiple agents are designed to manage IPS problem called I2 architecture where in two agent functions are used: Interface agent and Intelligent agent.
- Agents are intelligent to handle different cases accordingly. They would also learn from the environment.
- Agent based IPSM improves the performance of the system. It would also be able to analyze itself in terms of behaviour, error and success. The other performance measures are also taken in to account.
- Will be controlled and managed by control function.

This section begins with IPSM architecture followed by the multi agent architecture and experimental setup.

2.1 IPSM Architecture
The agent IPSM architecture is multi-agent architecture [4] has the I2 (I-Square) Functionality. Here two different agent functions are associated with Agent-Engine at different levels: Interface and Intelligent Agent [5]. The Interface agent [6] function is used to identify IPS processes from the user processes. The Intelligent agent function is responsible to manage the knowledge based of the system. The agent IPSM functions with the help of these two agent functions.

This agent IPSM is responsible to perform the following major tasks:

- Identify priorities of each process for each resource
- Maintain queue of process-resource-request

- Assigning TOKEN to that process that get chance to allocate the resource
- Transfer the control of a resource to the requested process
- change status of Resource
- change status of the process
- Allocation of resources
- Manage a TIMER to check the time stamp for each process allocated the resource. (to avoid deadlock and starvation)
- Lock and Release the resource (critical section management)
- Interfacing with user through messages
- Communication between agents using agent specification language (ASL).

The general functionality of agent IPSM is shown in figure-1. For proper functioning of the system following components of this system are used:

i) **Agent Engine:** The agent engine runs in background and acts as an interface agent between user and agent IPSM. All the processes are filtered here and the IPS processes are further transferred to agent IPSM for further solution.

ii) **Control Function CFAE (Control Function Agent Engine):** The control part of agent engine is the collection of Control Function which is responsible for controlling and managing the different modules. The control function P has two properties.

If, P=0 means negative clock pulse, then control will not be transferred
Otherwise, the control needs to be transferred

Here four major control functions are defined.
P1: To set and check the status of the resource
P2: To grant and release resources
P3: To manage the timer
P4: To update and access the knowledge base

iii) **Module PROCESS-RESOURCE-REQUEST:** This module will grant the permission to acquire and release the resources. A process to use the resource must first gain permission from the agent. Here we must guarantee that no process may starve. Here the agent will also maintain a queue of all the process requesting for acquire a resource. Queue would be maintained on the basis of FCFS strategy of scheduling.

iv) **Module STATUS management:** The resources are also replaced by the same agent who will treat like a resource manager which records the status of all of the resources. A process to get control of a resource has to get control from the resource manager. Once it is granted, the process can now use the resource.

v) **Module TOKEN management:** We also introduce the concept of TOKEN in the system which will be passed from one to another process those request to acquire the same resource.

vi) Module TIME management: A process may execute only if it is on the top of the queue, and if the permission is granted, resource is available and token is passed. Now the agent would also maintain the time stamp for each process so that the chances for deadlock or blocking of resources can be avoided.

2.2 Multi Agent Architecture

A Multi-Agent System (MAS) is a system [7] composed of multiple interacting agents. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. The agents in a multi-agent system as mentioned in figure 2 have several important characteristics:

- Autonomy: the agents are at least partially autonomous
- Local views: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge
- Decentralization: there is no designated controlling agent (or the system is effectively reduced to a monolithic system) [8].

3. AGENT INTER PROCESS SYNCHRONIZATION MANAGER

It has been shown in previous section that the agent based model is useful to design and implement recommended systems. In this section we present the Agent Inter process Synchronization Manager, is modeled using agent based approach. This section begins with the description of the algorithm suggested in section 3.1 which is then followed by its implementation using an application in section 3.2.

3.1 Design of the Agent

This section discusses the design of agent IPSM as a better solution for IPS problem. It also states that an Agent-Based IPSM in these kinds of problems can provide a valuable perspective over the more Non Agent-Based architectures. An Agent-Based IPSM can mirror to a large extent the way a distributed environment is organized in practice and can therefore support optimization of processes where this was not possible with Non Agent-Based solution [9].

However to develop an Agent-Based IPSM all these issues are addressed in relation to each other, which makes the design complicated. The solution aims at delivering a prototype system that can be used to assess both technical and organizational feasibility [10]. The algorithm is given to design the following modules:

- Module Agent_Engine
- Module AgentIPSM
- Module StatusManagement
- Module TokenManagement
- Module Process_Resource_RequestManagement
- Module KnowledgebaseManagement

3.2 Implementation

The agent based implementation is an attempt to experiment the algorithmic design. The experimental setup suggested here emphasizing on specific case which is the IPS situation of Windows Operating System.

The Inter Process Synchronization Manager suggested here is a plug-in for Windows' built-in Task Manager [11]. It expands the basic functionality and gives a powerful control over running processes. Inter Process Synchronization Manager can show process modules, process memory map, used kernel handles, opened files, file properties, and lots of other info. It shows more processes, adds lots of useful items to process the context menu, and adds a new "IPSM" submenu to the main window menu, and so on.

It is also providing an opportunity to use IPS Debugger to analyze and add processes facing IPS problem to its debugger and then synchronize internally using agent IPSM.

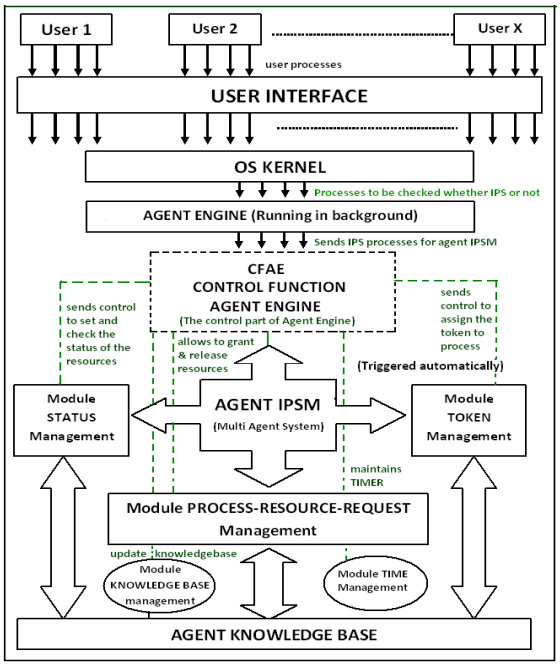

Figure 1: Agent IPSM Functionality

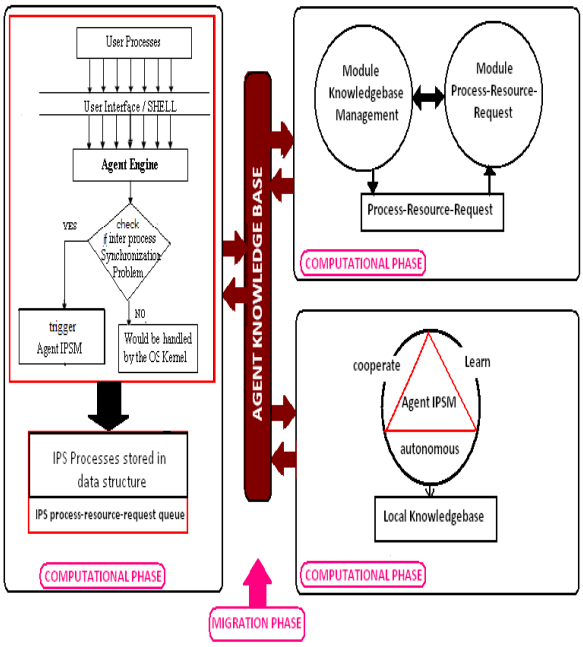

Figure 2: Multi-Agent Architecture of Agent IPSM

4. PERFORMANCE EVALUATION OF AGENT IPSM

The performance evaluation of the Agent IPSM is basically divided into two modules-

(1) Validation of the results
(2) Performance Evaluation.

To achieve the above said goals following are the sequence of jobs to be performed:

STEP # 1. Installation of the system which is described in Appendix B
STEP # 2. The system has many menu options; from these 'Test' option is to be selected.
STEP # 3. The test is performed on normal scenario which can be performed by selecting the option 'Normal' from 'Test' Menu.
STEP # 4. IPSM based testing performance is recorded by selecting the option "IPSM" from 'Test' menu.

The performance of the system which in this paper addressing as 'Normal mode' is the scenario where process manager of different Windows based operating system is used to record. This mode is the Task Manager of the operating system. This mode can also be viewed by selecting 'Normal' option of 'Test' Menu.

Another mode is 'IPSM mode' which is the IPSM process manager of the Windows based operating system.

This mode can be viewed by selecting 'IPSM' option of 'Test' Menu. The test has successfully been performed Windows XP based operating system.

4.1 Performance Evaluation Criteria

The test was conducted on following different parameters of a process out of which only five **parameters i, ii, iii, iv and v** were showing the significant change in output during testing at different time intervals. Rest of the parameters had the same output in all phases of testing after observation of different phases. Hence the test was more focused on first five parameters only.

i) CPU usage (in percentage): This parameter belongs to how much CPU is busy during execution of any process. Abbreviated as CU.
ii) CPU WAITING TIME (in seconds): How much time a process waits for CPU execution. Abbreviated as CW.
iii) MEM USAGE (in KB): How much memory space is used by a process. Abbreviated as MU.
iv) PAGE FAULT: Number of page faults occurred for a process. Abbreviated as PF.
v) TURN-AROUND TIME (in seconds): The time a process takes in execution. Abbreviated as TAT.
vi) VM SIZE (in KB): How much Virtual memory is consumed by a process. Abbreviated as VMS
vii) HANDLES: How many number of handles a process uses. Abbreviated as HD.
viii) THREADS: How many number of threads are used by a process. This case is specific to single user single threading system so that the no. of threads are one in all the phases. Abbreviated as TH.
ix) I/O READ: How many I/O reads performed during execution of a process. Abbreviated as IOR.
x) ST TIME (in hours:minutes:seconds): The start time of the process execution begins. Abbreviated as ST.
xi) END TIME (in hours:minutes:seconds): The time at which the process executes Abbreviated as ET.

4.2 Results

After evaluation of testing performed on normal mode and IPSM mode, following five statements state that agent IPSM is better solution as compared to normal case:

When a process is in running state, IPSM consumes more CPU over normal mode of execution …….Statement [i]

IPSM allows only one process at a time to remain in critical section…….....Statement [ii]

IPSM states that more memory is used for the process in critical section ……....Statement [iii]

IPSM states that when a process is in critical section, more page fault occur ……...Statement [iv]

IPSM consumes lesser turn-around time as compared to normal mode ……...Statement [v]

Above statements are further shown in figure 3 through the following comparative analysis:

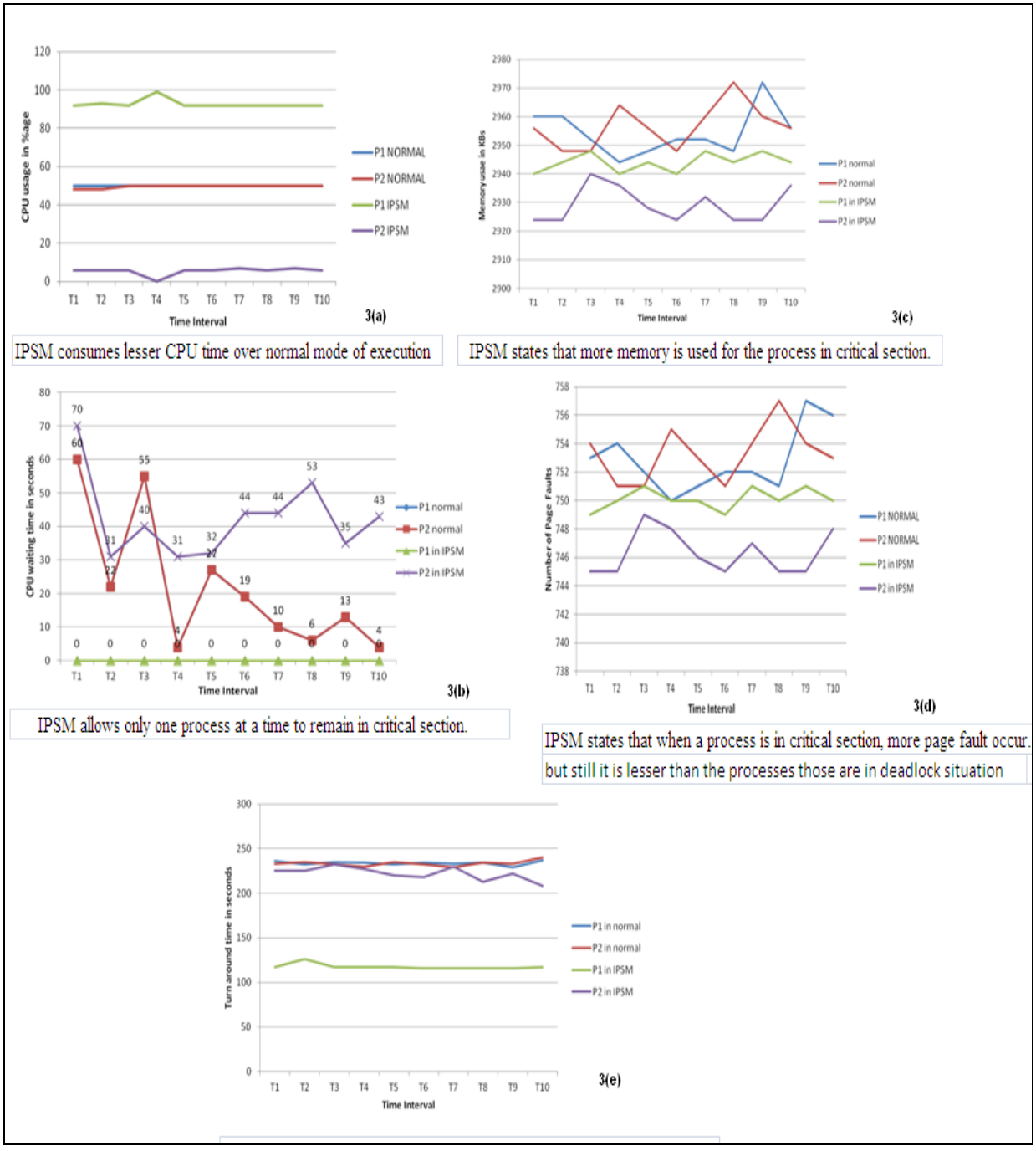

Figure 3: Comparative Analysis of Normal and IPSM mode

Statement [i]: When a process is in running state, it consumes more CPU over normal mode of execution

Proof of statement [i]: Observations from graph shown in figure 3(a) prove that statement [i] is correct.

i) Process P1 and P2 in normal execution mode have approximately same percentage of CPU usage over a period of time.
ii) Process P1 and P2 are in using same amount of CPU.
iii) Process P1 running in IPSM used maximum percentage of CPU usage that shows that process P1 is in critical section and is in execution.
iv) Process P2 running in IPSM used minimum percentage of CPU usage that shows that P2 is ready for execution.
v) Process P1 is in running state.

Hence, proved that *"When a process is in running state, it consumes more CPU over normal mode of execution"*

Statement [ii]: IPSM allows only one process at a time to remain in critical section.

Proof of statement [ii]: Observations from graph shown in figure 3(b) prove that statement [ii] is correct.

i) Process P1 and P2 are in deadlock condition hence, they are circularly waiting for CPU.
ii) Process P2 have more waiting time as it is in ready state.
iii) Process P1 have no waiting time as it is in execution.
iv) Process P1 is only in critical section.

Hence, proved that "*IPSM allows only one process at a time to remain in critical section."*

Statement [iii]: IPSM states that more memory is used for the process in critical section.

Proof of statement [iii]: Observations from graph shown in figure 3(c) prove that statement [iii] is correct.

i) Process P1 and P2 in normal mode are not in critical section hence using more memory.
ii) Process P2 in IPSM is not in critical section hence doesn't need more memory.
iii) Process P1 in IPSM, is in critical section hence using more memory as compared to process P2 in IPSM.

Hence, proved that "*IPSM states that more memory is used for the process in critical section."*

Statement [iv]: IPSM states that when a process is in critical section, more page fault occur.

Proof of statement [iv]: Observations from graph shown in figure 3(d) prove that statement [iv] is correct.

i) Number of page faults occurred in process P1 and P2 in normal mode are more, as both are in deadlock situation.
ii) Process P2 in IPSM is not in critical section hence, lesser number of page faults occurred.
iii) Process P1 in IPSM is in critical section hence, takes more number of page faults in execution state.
iv) Process P1 and P2 in IPSM are comparatively consuming lesser number of page faults than processes in deadlock.

Hence, proved that "*IPSM states that when a process is in critical section, more page fault occur."*

Statement [v]: IPSM consumes lesser turn-around time as compared to normal mode

Proof of statement [v]: Observations from graph shown in figure 3(e) prove that statement [v] is correct.

i) Process P1 and P2 in normal mode consume maximum turn-around time as these processes are in deadlock.
ii) Process P2 in IPSM consumes more time from Process P1 in IPSM as it is in ready state.
iii) Process P1 in IPSM consume lesser time overall as compared to all the processes whether in ready state or in deadlock.

Hence, proved that *"IPSM consumes lesser turn-around time as compared to normal mode"*

After observing and proving all the statements, it implies that ***CPU is showing comparatively better performanace in IPSM mode than normal mode when evaluated for Windows based operating system.***

In other words, we can say that ***for the problem of Inter-process synchronization IPSM suggests a better solution for Windows based operating system in IPSM mode as compared to normal mode of execution.***

5. CONCLUSION & FUTURE SCOPE

The agent-based design method presented in this thesis helps the designer to solve IPS problems, by offering concepts to structure the problem and to organize the solution. The design method encourages to develop local solutions and to reason about their dependencies.

The presented agent-based design framework IPSM is aimed at developing solutions that run on single processors. The future scope includes the design framework on multi processor system to solve the same problem domain.

Here the theoretical foundations of an agent-based design method for IPS problems were presented. During the research .NET framework was used. One of the disadvantages of using Visual Studio.NET and the .NET framework to develop applications has been the lack of cross-platform support. Hence the future scope would be testing the system on Linux/Unix based operating system.

6. REFERENCES

[1] Wallis, P. Ronnquist, R. Jarvis, D. Lucas, "The automated wingman - Using JACK intelligent agents for unmanned autonomous vehicles", Aerospace Conference Proceedings, 2002. IEEE , Agent Oriented Software, Carlton, Vic., Australia, Volume: 5, pp 5-2615- 5-2622 vol.5, 2002

[2] U. Ramachandran, M. Solomon, M. Vernon, "Hardware support for interprocess communication", Proceedings of the 14th annual International Symposium on Computer architecture. Pittsburgh,

Pennsylvania, United States. Pages: 178 - 188., ISBN 0-8186-0776-9 , 1987

[3] Hyacinth S. Nwana, "Software Agents: An Overview", Intelligent Systems Research Advanced Applications & Technology Department ,BT Laboratories, Springer Berlin / Heidelberg, pp 59-78, 1997

[4] Acebo, E. and de la Rosa, J. L., A Fuzzy System Based Approach to Social Modeling in multiagent Systems. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Palazzo Re Enzo, Italy, 2002.

[5] Amgoud L. and Kaci S., On the generation of bipolar goals in argumentationbased negotiation. In (I. Rahwan et al, eds.) Proceedings of the 1st Int. Workshop on Argumentation in Multi-Agent Systems (ArgMAS), volume 3366 of LNCS, 192-207. Springer, Germany, 2004.

[6] R. Belecheanu, S. Munroe, M. Luck, T. Payne, T. Miller, M. Pechoucek, and P. McBurney. Commercial applications of agents: Lessons, experiences and challenges. In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems, Hakodate, Japan, pages 1549--1555, 2006. ACM Press.

[7] Giuseppe De Giacomo, Yves Lesperance, and Adrian R. Pearce. Situation Calculus-Based Programs for Representing and Reasoning about Game Structures. In Fangzhen Lin Sattler and Ulrike, editors, Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010), Toronto, Canada, pages 445--455, 2010.

[8] Blom, L., Michelle and Adrian R. Pearce. An Argumentation-Based Interpreter for Golog Programs. In International Joint Conference on Artificial Intelligence (IJCAI 2009), Los Angeles, pages 690--695, 2009.

[9] Li Xining, Comparison of communication models for mobile agents, Systemics, Cybernetics And Informatics, Volume 1 - Number 2

[10] Miller, T. and McBurney, P.. On illegal composition of first-class agent interaction protocols. In *Proceedings of the Thirty-First Australasian Computer Science Conference*, volume 74 of *CRPIT*, pages 127--136, 2008. ACS.

[11] Mobach D.G.A., Overeinder B.J., and Brazier F.M.T., A WS-agreement based resource negotiation Framework for mobile agents, Scalable Computing: Practice and Experience, 7 (1):23 – 36, 2006.

[12] Shehory Onn, Sycara Katia, Sukthankar Gita, Mukherjee Vick, Agent aided aircraft maintenance, School of Computer Science, Carnegie-Mellon University

[13] Albert J.N. van Breemen, Agent-Based Multi-Controller System, PhD Thesis, University of Twente, Twente University Press, 2001

[14] Wooldridge M and Jennings N. R., Intelligent Agents: theory and practice. The Knowledge Engineering Review, 10(2), 115-152, 1995.