Intelligent Software Agent Technology: An Overview

G. S. Bhamra  
DCSE, M. M. University, Mullan-133203, Haryana, India

A. K. Verma  
DCSE, Thapar University, Patiala-147004, Punjab, India

R. B. Patel  
DCSE, G.B. Pant Engineering College, Ghurdaurl-246194, Uttarakhad, India

ABSTRACT
Mobile Agent is an autonomously transportable code migrating itself from one node to another in a heterogeneous network without losing its operability. They have become commercially practicable with recent technologies and have the potential for revolutionizing distributed and network applications more recently in wireless sensor networks and bioinformatics. The main aim of this study is to motivate the researchers into the field of intelligent software agent technology by providing an overview and updated comparison of the current mobile agent platforms.

General Terms
Distributed Computing, Agent Technology.

Keywords
Intelligent Agent, Mobile Agent, Mobile Agent Platforms.

1. INTRODUCTION
Intelligent software agent technology is an interdisciplinary technology inherited from different research disciplines such as, Distributed Computing (DC), Distributed Artificial Intelligence (DAI), advanced knowledge base systems, and human computer interaction. Intelligent agents are the autonomous software objects, which have access to geographically distributed and heterogeneous information resources. The motivating idea of this technology is the development and efficient utilization of such agents as they simplify the complexities of DC [1],[2],[3]. Agent research stems from the work in DAI conducted in the 1970’s. Carl Hewitt proposed an Actor system [4] where each Actor had an explicit internal state and had the capability to respond to the messages of other Actors. The last decade has seen a huge expansion of agent systems to solve practical problems in various domains. The next generation of distributed applications demand a large number of interacting components in the form of services or intelligent agents[5].

2. TAXONOMIES AND CHARACTERISTICS
A number of classifications of agents and intelligence criteria have been proposed in the literature. Software Agents with different attributes are being designed depending upon the goals to be achieved and responsibilities to be carried out. Gilbert et al. (1995)[6] described intelligent agents in terms of the three dimensions of agency, intelligence, and mobility. Nwana (1996)[1] identified three primary attributes, autonomy, cooperation, and learning and classified the intelligent agents as collaborative, collaborative learning, interface, and smart. Based upon the mobility, reactivity, role played, hybrid philosophies and secondary attributes, such as versatility, kindness, reality, trustworthiness, temporal continuity, ability to fail gracefully, and mentalistic and emotional qualities, agents can be classified as mobile agents, information/internet agents, reactive agents, and hybrid agents [7],[8],[9]. Several definitions for the agents have been proposed suggesting a set of defining characteristics that every agent must possess. These capabilities are listed below:

- **Autonomous:** Autonomy is the ability of operating without the intervention of users. The degree of autonomy and authority vested in an agent is called its agency. It can be measured qualitatively by the nature of the communication between the agent and other entities involved. At a minimum, an agent must run asynchronously. The degree of agency is enhanced if an agent represents a user/human in some way. Human behavior and practical reasoning in agents can be incorporated using Belief-Desire-Intension (BDI) model of agency in terms of ‘mentalistic’ notions [10]. The origins of the model lie in the theory of human practical reasoning developed by the philosopher Michael Bratman [11].

- **Reactive or Deliberative:** the ability of perceiving any change in the environment and suitably reacting to it.

- **Adaptive:** the ability of sensing the environment and reconfiguring in response dynamically. This can be achieved through the choice of alternative problem solving rules/strategies or algorithms.

- **Pro-activeness:** the ability of exhibiting goal-directed behavior in order to satisfy a design objective.

- **Intelligence:** the ability to accept the user’s statement of goals and carry out the task delegated to it. It is the degree of reasoning and learned behavior.

- **Temporal continuity:** the ability of persistence of identity and state over long periods of time.

- **Social, Cooperative and Collaborative:** the ability of interacting and negotiating with other agents in groups and/or human users to achieve a common goal via some kind of agent communication language. They cooperate, negotiate, and collaborate with other agents to make a smart system.

- **Flexibility:** the ability of exhibiting reactivity, pro-activeness and social ability simultaneously.

- **Mobility/Migration/Navigation:** the ability of migrating from a node to node in a heterogeneous network. Mobility can be of weak and strong in nature. Weak mobility moves an agent who is currently idle (not running) to another machine, i.e., only agent code is transferred and the agent starts its execution from the starting point on the remote machine. Strong mobility moves an agent who is currently performing its task (running) to another machine, i.e., agent code as well execution state (data stack) is transferred and the agent...
continues its calculations/execution from the point of suspension on the remote machine like it was never interrupted. The basic aim is to perform access and information processing locally to a resource. On the other hand, mobility raises security and efficiency issues.

3. MOBILE AGENT

An intelligent agent with mobility feature is known as Mobile Agent (MA). MA is an autonomously transportable code migrating itself from one node to another in a heterogeneous network without losing its operability. In other words, program running at a host can suspend its execution at an arbitrary point, transfer itself to another host (or request the host to transfer it to its next destination) and resume execution from point of suspension. When the MA reaches a server, it is delivered to an agent execution environment (agent platform). Then, if the agent possesses necessary authentication credentials, its executable parts are started. To accomplish its task, the MA can transport itself to another server in search of the needed resource/service, spawn new agents, or interact with other stationary agents. Upon completion, the MA delivers the results to the sending client or to another server. Once it is launched by a user, it can travel from node to node autonomously, and can continue to function even if the user is disconnected from the network. Various attributes of a MA are i) Identification to identify a MA and its dispatching station, ii) Itinerary consisting of the number and order of the hosts to be visited by MA, iii) Data unit containing the required data, iv) Code unit containing the transportable code, v) Execution state, and vi) External state for intermediate results [1],[3],[7],[12],[13],[14],[15],[16].

3.1 General benefits of MAs

MAs can be viewed as a replacement, refinement or extension of the traditional client/server paradigm. They are naturally heterogeneous and goal oriented in nature and may be launched into the unstructured network and roam around to gather information. MAs provide the following key advantages:-

- **Reduction in network bandwidth and latency:** MAs allow efficient and economical use of communication channels having low network bandwidth and high latency (response time delay). With MA, a single serialized object is transmitted over the network carrying the small amount of resultant data only, reducing the consumption of network bandwidth and latency.

- **Reduction in network traffic:** They reduce the network traffic because only resultant data is carried over the network to a central site and unnecessary intermediate results transmission is avoided.

- **Robust and fault-tolerant:** MA may have the intelligence to bypass the faulty hosts or seek temporary shelter on a reliable host. MAs are especially appropriate for wireless network links because the possibility of failure due to network faults is reduced since the MA needs only to be transported once between two sites.

- **Disconnected operation:** MA is not tightly bound to its dispatching host. It can continue functioning even if it is disconnected from its “home”, and send back results upon reconnection. This feature is particularly useful for low-cost, light weight, portable computing devices such as Personal Digital Assistants (PDAs) having the low processing powers, memory constraints, and intermittent low bandwidth connection to the main network.

- **Load balancing:** Some of the processor and memory intensive tasks of a program in a lightweight mobile device can be offloaded to other more powerful and not heavily loaded platforms due to agent’s strong mobility feature.

- **Scalable, configurable and re usable:** A distributed application, built as a collection of mobile agents, can be easily extended with new functionality in the form of agents with slight modifications to the master agent. Number of participating hosts can be increased without any significant impact on the complexity of the application. MAs are self-contained and highly re usable.

- **Cloning:** The parent agent can also clone several child agents to implement concurrent operations, and improve the efficiency.

- **Rapid prototyping:** Software components can be flexibly and dynamically deployed in the form of MAs. This facilitates the rapid prototyping of distributed applications.

4. AGENT TECHNOLOGY STANDARDS

There are currently a wide range of different agent architectures, frameworks and systems developed for both research and industrial purposes. To unify these approaches, three standardization efforts have appeared with the overall aim of increasing interoperability between agent systems and even dealing with the problem of misuse involving MAs.

- **MASIF:** The Mobile Agent System Interoperability Facility (MASIF) has been in development by the Object Management Group (OMG) since 1995 to promote interoperability and mobility among agent platforms [17]. MASIF uses a procedure-oriented interaction model using Remote Procedure Calls (RPC) or Remote Method Invocation (RMI) technology.

- **KQML:** The Knowledge Query Meta Language (KQML) [18] is one of the most popular and widely used protocols for defining agent-to-agent communication. KQML is the oldest project, developed in 1992 by the DARPA Knowledge Sharing Effort consortium.

- **FIPA:** Foundation for Intelligent Physical Agents (FIPA) is a non-profit organization created in 1996 aimed at developing software standards (specifications) for maximizing interoperability within and across agent based systems. FIPA-ACL (Agent Communication Language) is adopted and used by many parties as their agent communication language. ACL uses KIF (Knowledge Interchange Format) and KQML to communicate knowledge and queries to others [19]. Three mandatory roles are present into a FIPA compliant agent platform: AMS (Agent Management System) that controls access and use of the platform, DF (Directory Facilitator) that provides yellow pages service and ACC (Agent Communication Channel) that provides the Message Transport Service for FIPA ACL messages delivery between agents living into different agent platforms. In order to promote interoperability between agent platforms, a number of standard Message Transport Protocols (MTPs) and MTP interfaces have been defined by FIPA. FIPA does not define nor require
5. WHY JAVA?
The Java language has several inherent advantages that make it an ideal language for implementing agents: multithreading support, portability due to small size of compiled byte code and Java Virtual Machine (JVM), ease of network programming constructs (sockets), migration and communication support using object serialization and de-serialization [20], dynamic class loading and remote method invocation (RMI), highly secure program execution due to constantly growing security model (security managers, security policies, access to multiple databases over the Web using simple database connectivity interface (JDBC). In addition the JavaBeans specification enables the creation of reusable Java components with well-defined interfaces and behaviors.

6. APPLICATIONS OF AGENTS
The qualitative strengths of intelligent autonomous agents make current agent platforms an attractive choice for a wider range of distributed applications. Today, agents are deployed in different settings and application domains such as Information Retrieval System (IRS); Systems and Network management; Mobile Computing; Managing Mails and Messaging; Workflow and Administrative Management; Electronic Commerce; Industrial control; Internet searching; Personal assistance; Games; Software distribution; Bioinformatics; Clinical Data analysis for medical diagnosis; Distributed Data Mining (DDM); Intrusion Detection System (IDS); Peer-to-Peer Computing; Grid and Cluster Computing; Wireless Sensor Network (WSN); Unmanned Air and Under-Water Vehicles; Oil production systems etc.

7. MOBILE AGENT PLATFORM
A Mobile Agent Platform (MAP) a.k.a. Agent Execution Environment (AEE) or Agent Development (Tool)kit, is a middleware, distributed, server application that provides the appropriate functionality to MAs to authenticate, execute, communicate (with other agents, users, and other platforms), migrate to other platform, and use system resources in a secure way. A Multi-Agent System (MAS) is distributed application/system composed of multiple interacting intelligent agent components. The main purpose of a MAP is to provide a framework with higher level of abstraction for the development, execution and management of MAS without going into the low level communication details and just focusing on the desired functioning of the agents themselves [9],[21],[22],[23].

The conceptual model of a java based MAP is depicted in Figure 1. It consists of a) Host machine or a Network Node with Operating System and other hardware; b) Java Virtual Machine (JVM)-a runtime environment for platform independence and security; c) Mobile Agent Platform (MAP)- A multithreaded server application running on the top of JVM and provides the agent API with the basic functionality for agent management, migration, communication and security etc., in the form of java packages; and d) Communication Infrastructure (wired or wireless network) connecting the Host machines. Java MAs are special Java objects, executed as threads within an AEE. TCP/IP is used as the main transport mechanism.

7.1 Basic services provided by a MAP
The behavior of a MAP is based on the services it offers. MAPs provide a set of core services and could be enriched with new services, implemented as service agents on top of the core functionality. These properties can be functional and non-functional. The basic facilities provided by an AEE include the following:

- **Mobility:** Mobility mechanisms include remote invocation, itinerary management, cloning, and programming language support. Most platforms use application protocols on top of TCP for transport of agent codes and states. Java based platforms use remote method invocation (RMI), TCP Sockets, object serialization, and reflection.

- **Communication:** A communication model allows the agents to communicate with each other and the system to control agents. Most systems implement a distributed event communication and/or a message-passing mechanism using Java objects. Sockets are a low-level communication service which allows programs to exchange information. Java TCP socket classes are an abstraction of TCP connection oriented sockets. Remote Procedure Call (RPC) is a high-level communication mechanism. Java RMI is a type of RPC mechanism. RMI can be viewed as a newer object-oriented version of SunRPC. Communication performance of a MAP can be measured using Round trip time (RTT).

- **Security:** The Security model of an AEE handles the security problems, i.e., how to protect the AEE from malicious agents, agents from a malicious AEE, one
agent from another, the communication between AEEs, and the host machine from the AEE. A variety of security mechanisms exist including encryption, public key infrastructure (PKI), and support for executing digitally signed code. PKI serves as a foundation for mobile agent security services and makes authentication, non-repudiation, and encryption readily available to agent developers and users. X.509 certification and Pretty Good Privacy (PGP) or the Secure Socket Layer (SSL) protocols represent two different approaches to the distribution of trust on the Internet. Digital signature also serves as a means of confirming the authenticity of an agent. Typically the code signer is either the creator of the agent, or the user of the agent. X.509 is based on the concept of Certification Authority (CA) server, which is a centralized control of trust to certify and manage all the certificates, while PGP is totally decentralized but very difficult to scale up the whole system. Traditional security mechanisms rely on cryptographic techniques to implement authentication, authorization, and access control. In most of java based systems, the designers choose to rely on the Java security model and/or provide customized extensions to this model. Security is an ongoing research issue and should be regarded as an ongoing feature. MA architecture should be designed a way so that new and better security mechanisms can easily be incorporated as they became available.

- **Identification/Naming Scheme**: Agents must be uniquely identified. Proper identification allows control, communication, cooperation, and coordination of agents to take place. All schemes used in the current systems are variations of generating a unique number sequence identifying each created agent.

- **Portability/Platform Heterogeneity**: All current MA systems must deal with porting agent code and data to work on multiple platforms. Before Java, systems such as Agent Tcl and ARA depended on an operating system specific and language specific interpreters for agent code and data. The problems of these approaches are performance and scalability. The Java virtual machine (JVM), although represents a better compromise because platform-neutral byte code can be compiled just in time to adequately improve the performance problems.

- **Resource Management**: Resource management component deals with the fair distribution of the access to low-level system resources such as CPU cycle, disk, memory, graphic subsystem and network among requesting agents. Agents may also require higher-level access such as persistence service (i.e., database), thread, and services from static entities (i.e., back-end directory server, SQL server).

- **Resource Discovery**: Resource discovery covers an area complementary to resource management. When an agent arrives at a site, it should be able to discover the services offered at that site. It would be much better if schemes were designed as part of the overall architecture to allow both prior and just-in-time discovery of services.

- **Control/Computation**: The basic goal of control is to provide the ways in which an agent may be created, started, suspended, stopped, duplicated, or instructed to self-terminate and how these lifecycle activities of a MA can be coordinated. The common approach is to provide these abilities in the root MA class. Computations are essentially organized in a “jump-act-jump” pattern.

- **Persistence**: The basic goal of persistence or data management is to save the agents and its associated data onto secondary storage for fault tolerance or other purposes. Data management is important to allow an MA system to scale and also affects performance.

- **Interoperability**: It is the ability of a system to work with or use the parts or equipment of another system. An AEE provides interoperability if it has compliance with certain standards like FIPA, OMG’s MASIF etc.

- **Performance**: The performance of a multi-agent system is a measure using a set of statistical indicators of the system’s major outputs and its consumption of resources, where typical indicators include throughputs, response time, number of concurrent agents/tasks, computational time and communications overhead [24].

- **Scalability**: A system is said to scalable if its capacity to do useful work increases with the increase in the size of the system [24]. A MAS can be easily extended with new functionality in the form of agents with slight modifications to the master agent. Number of participating hosts can be increased without any significant impact on the complexity of the application. Scalability of a MAP can be measured as a performance degradation of the individual agent with the addition of extra nodes and extra agents in the agent society.

- **Stability**: Stability is a property of equilibrium. A MAS is in equilibrium when the statistical properties of its performance indicators remain stationary for a given variation in the system’s external load. Thus, instability occurs when a small perturbation of some system parameter leads to a sharp and persistent deviation in the system’s performance indicators [24].

8. **QUALITATIVE COMPARISON OF IMPORTANT MAPS**

Various MAPs have been developed both as academic research projects and commercial products by industries, with some platform projects abandoned altogether. These platforms can be broadly categorized as Java and non-Java based. Non-java based systems are operating system services accessible via a scripting language with interpreter (e.g., Tcl/Tk) and runtime support. Some of the important classical platforms that appeared in the literature during 1995 to 1999 are Agetls [25],[26], ARA [27], Ajanta [28], Concordia [29], D’Agents /AgentTcl [30],[31], Grasshopper [32], Gypsy[33], Hive [34], Jackal [35], JAMES [36], MOLE [37], Soma [38], TACOMA [39],[40], and ZEUS [41].Other important current platforms are JAPL Platform [42], AAP[43], ABLE [44], ADK [45], AgentBuilder [46], Agentdock [47], AgentScape [48], CAPA[49], CapNet [50], Jack [51], C-BDI [52], Cougaar [53],[54], CybeelePro [55], DECAF [56], FIPA-OS [57],[58],[59], Genie [60], IMAGO [61],[62], JADE [63],[64],[65], Jason [66], MadKit [67], Mage [68], MobileSpaces [69],[70], PMADE [13],[14], Sage [71], SEAGENT [72], Semou [73], SHIP-MAI [74], Spade [75], SPRINGS [76], Tryllian’s ADX [77] and Voyager [78].
An exhaustive study of each MAP is not feasible due to the large number in existence. So, qualitative comparison of some prominent current MAPs is provided here and Table 1 and Table 2 shows the results of this study of current commercial as well as academic MAPs taking into account some of the features they provide. The names used here for MAPs and standards are the registered trademarks of their respective developers/organizations. The features include the following fields:

- **CP** field indicates whether a MAP is a commercial product (‘C’) or an academic research project (‘A’).
- **Lang** field shows the language in which each MAP is implemented.
- **OS** indicates whether a MAP is open-source or not.
- **Std** indicates whether a MAP provides interoperability with other platforms or technology for which it should be in compliance with the agent technology standards such as FIPA, OMG-MASIF, etc.

- **Sec** points out whether security is taken into account.
- **Communication** indicates which communication technology each MAP uses such as RMI, TCP/IP etc.
- **Mob** indicates which mobility model is used, i.e., strong or weak.
- **GUI** is for the Graphical User Interface feature of the MAP.
- **BDI** points out the support for Belief-Desire-Intension model of practical reasoning.
- **CS** points out the Current Status (whether it is currently maintained and what is the last available version).
- **Use** indicates use of MAP in practical applications, development projects, case studies etc.
- **Miscellaneous** indicates any other additionally highlighted feature provided by MAP.

### Table 1. Qualitative comparison of MAPs

<table>
<thead>
<tr>
<th>Features Platform ▼</th>
<th>CP</th>
<th>Lang</th>
<th>OS</th>
<th>Std</th>
<th>Sec</th>
<th>Communication</th>
<th>Mob</th>
<th>GUI</th>
<th>BDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>3APL</td>
<td>A</td>
<td>Java</td>
<td>Yes</td>
<td>FIPA</td>
<td>No</td>
<td>RMI</td>
<td>weak</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ABLE</td>
<td>C</td>
<td>Java</td>
<td>Yes</td>
<td>FIPA</td>
<td>Yes</td>
<td>RMI, HTTP, Data Flow, Events</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ADK</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>asynchronous, synchronous, Jini, JavaSpaces, RMI</td>
<td>weak</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AgentBuilder</td>
<td>C</td>
<td>Java</td>
<td>No</td>
<td>FIPA</td>
<td>Yes</td>
<td>TCP/IP, KQML</td>
<td>weak</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AgentScape</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>FIPA</td>
<td>No</td>
<td>SunRPC, HTTPS</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CAPA</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>FIPA</td>
<td>No</td>
<td>synchronous, RMI</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CapNet</td>
<td>A</td>
<td>C#</td>
<td>No</td>
<td>FIPA</td>
<td>Yes</td>
<td>MTS, XML-SOAP, HTTP, DCOM, IIOP, SMTP</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>C-BDI</td>
<td>C</td>
<td>C++</td>
<td>No</td>
<td>FIPA</td>
<td>Yes</td>
<td>TCP/IP</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cougaar</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>RMI, CORBA, HTTP</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CybelePro</td>
<td>C</td>
<td>Java</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>RMI</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DECAF</td>
<td>A</td>
<td>Java</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>KQML</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>FIPA-OS</td>
<td>C</td>
<td>Java</td>
<td>Yes</td>
<td>FIPA</td>
<td>MASIF</td>
<td>Yes</td>
<td>ACL, CORBA-IOR, IIOP, RMI, XML</td>
<td>weak</td>
<td>Yes</td>
</tr>
<tr>
<td>IMAGO</td>
<td>A</td>
<td>Prolog</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>messenger imago(agent)</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Jack</td>
<td>C</td>
<td>Java</td>
<td>No</td>
<td>FIPA</td>
<td>Yes</td>
<td>TCP/IP, Event-Plan</td>
<td>weak</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JADE</td>
<td>C</td>
<td>Java</td>
<td>Yes</td>
<td>FIPA</td>
<td>Yes</td>
<td>synchronous, asynchronous, Events, plugin MTP for RMI, IIOP, HTTP, WAP</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Jason</td>
<td>A</td>
<td>AgentSpeak, Java</td>
<td>Yes</td>
<td>FIPA</td>
<td>No</td>
<td>synchronous, asynchronous, speech-act, KQML</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MadKit</td>
<td>A</td>
<td>Java</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>asynchronous, Java Sockets</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Mage</td>
<td>A</td>
<td>Java</td>
<td>Yes</td>
<td>FIPA</td>
<td>No</td>
<td>RMI</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MobileSpaces</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>TCP, UDP, SMTP</td>
<td>weak</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PMADE</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>MASIF</td>
<td>Yes</td>
<td>Java TCP Sockets</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SHIP-MAI</td>
<td>A</td>
<td>Java</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>RMI, ADC</td>
<td>weak</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Spade</td>
<td>A</td>
<td>Python</td>
<td>Yes</td>
<td>FIPA,</td>
<td>Yes</td>
<td>Jabber, XMPP, HTTP, ACL</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Platform</td>
<td>Features</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3APL</td>
<td>CS</td>
<td>3APL-M version for J2ME enabled mobile devices, last update on 19 November 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>Universities Research Projects; mobile devices</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>First platform to support cognitive agents</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABLE</td>
<td>CS</td>
<td>ABLE ver. 2.0; new features are continuously incorporated into the platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>agent-based modeling and simulation studies; system administration (IBM eServer iSeries); server diagnostics application; Feedback control system based on ABLE Autotune agents for automatically tuning the Apache Web server parameters; E-Commerce.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Eclipse based light weight framework with JavaBeans/AbleBeans components; reusable; scalable; java serialization for persistence; ABLE Rule Language (ARL) for rule and policy-based knowledge representation. Agent Naming Service for identification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADK</td>
<td>CS</td>
<td>not maintained further</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>air-ticket trading case study</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>agent-oriented G-net model(a type of high-level Petri net) to define the agent structure, agent behavior, and agent functionality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentBuilder</td>
<td>CS</td>
<td>AgentBuilder Lite and AgentBuilder Pro</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>agent-based modeling and simulation studies; eBusiness applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>IDE consisting of Project Manager, Ontology Manager,Agency Manager,Agent Manager and Protocol Manager for quickly and easily constructing intelligent BDI agents; Project Accessory Classes (PACs) contain the problem-specific code in Java, C; C++ libraries</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentScape</td>
<td>CS</td>
<td>AgentScape 2 Milestone 5 Release July 2013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>scalable, fault tolerant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAPA</td>
<td>CS</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>An extension of Mulan (Multi Agent Nets) architecture; Reference Nets(a special kind of high-level Petri nets) and Concurrency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CapNet</td>
<td>CS</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>Contingency Management Simulation System (CMSS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Component Agent Platform based on .NET; policy manager and agent credentials for security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-BDI</td>
<td>CS</td>
<td>a full DO-178B Level A certification package is planned for integration into any system’s safety case</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>capable of executing a variety of decision-making systems in the aerospace, oil &amp; gas and medical industries, and intelligent motor vehicle applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>A certifiable BDI agent platform to easily port JACK-based applications to the new platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cougaar</td>
<td>CS</td>
<td>latest release Cougaar v12.7 (30-Jul-2012)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use</td>
<td>Critical Infrastructure Modeling and Simulation; CougaarME for robots in WAHN; DARPA UltraLog program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Scalable; configurable; logical data model (LDM);survivable; heterogeneous; Java Security Manager and SSL for security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CybelePro</td>
<td>CS</td>
<td>CybelePro v. 3.1.3 is available as 3,6,10 and more than 10 nodes paid packages. Other versions of this product are OpenCybele v. 1.2.0(open source), CybeleLite v. 3.1.3(free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>License/Use</td>
<td>CS</td>
<td>Miscellaneous</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECAF (N/A)</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIPA-OS (N/A)</td>
<td>Standard FIPA-OS, MicroFIPA-OS</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGO (N/A)</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jack (N/A)</td>
<td>CoJACK (JACK with cognitive architecture) enables the creation of principled models of human behaviour.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JADE (N/A)</td>
<td>Workflows and Agents Development Environment (WADE) is a JADE extension and WOLF is an Eclipse plugin; Lightweight Extensible Agent Platform (LEAP) package runs on Java ME enabled wireless devices</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason (N/A)</td>
<td>Gold Miners, Electronic Bookstore</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MadKit (N/A)</td>
<td>Last 5th stable version MaDKit-5.0.3.1 on November 13 2013</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mage (N/A)</td>
<td>Agent-oriented Unified Model Platform (AUMP) modeling tool for system analysis and design stages; VASTudio development tool for system design, development and deployment stages (design module, programming module, runtime platform and toolkits)</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobileSpaces (N/A)</td>
<td>Reusable mobile agents for network management</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMADE (N/A)</td>
<td>PMADE version 1.0</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIP-MAI (N/A)</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this sense, Pham and Karmouch (2002) also extended their work and test the JADE messaging service. Chmiel et al. (2004)[90] also extended their work and test the JADE messaging, agent creation and migration services. In Spamming test, User agents are flooded with ACL messages by the Spammer Agent. In database access test, SQLAgents provides documents requested by a user agent. In communication evaluation, they compared the performance of existing MAPs and such comparisons provide an overview of the different alternatives. Most of these studies highlight a lack of performance, security, communication, and scalability.

In one of the earlier studies Pham and Karmouch (1998)[12] compared Aglet, Agent Tel, ARA, Concordia, MOLE, Odyssey, SACMA, SHIP-MAI, and Voyager and presented an overview of these platforms in the context of telecommunications. They compared the MAPs with the criteria of security, portability, mobility, communication, resource management, resource discovery, identification, control, data management and case study in telecommunications. Silva et al. (2000)[84] compared the performance and the robustness of eight Java-based Mobile Agent systems: Aglets, Concordia, Voyager, Odyssey, Jumping Beans, Grasshopper, Swarm and James by experimental study of benchmarking. They found James to be best in terms of performance and robustness as compared to others.

Ricordel and Demazeau (2000)[85], in a survey on multi-agent platforms with a particular focus on methodology, evaluate AgentBuilder, Jack, Madkit, and Zeus platforms with the criteria of analysis, design, development and deployment stages of software engineering with the qualitative values of Completeness, Applicability, Complexity, Reusability. Silva et al. (2001) [86] compared Telescript, Aglets, fJMAIN, D’Agents and AgentSpace MAPs according to the terminology, concepts and architecture proposed as part of the reference model. Criteria included in their reference model are: execution, management of agent types, management of identifiers, persistence, navigation, communication, interaction with external resources, and security. Camacho et al. (2002)[87] shows the performance of Zeus, JADE and SkeletonAgent MAPs when a MAS is composed by a several web agents is launched. This MAS provides documents requested by a user agent. In this sense, the authors measure the number of requested documents per time. Therefore, the conclusions are only valid for this MAS. Vrba (2003)[88] presents an evaluation of the messaging service performance of JADE, FIPA-OS, Zeus, and Jack. From the tests presented in this paper the authors conclude that JADE provides the most efficient messaging service and Zeus the worst one. However, the design reasons that cause this performance are not given and the implementations of the messaging service for each MAP are not detailed.

<table>
<thead>
<tr>
<th>Use</th>
<th>Mobile Agent based Software Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misc.</td>
<td>Agent Domain Controller(ADC); IAIK-JCE v2.5, X.509 certificates and RSA for security and authentication</td>
</tr>
<tr>
<td>Spade</td>
<td>Spade2</td>
</tr>
<tr>
<td>Use</td>
<td>N/A</td>
</tr>
<tr>
<td>Misc.</td>
<td>XML Router as a Message Transport System (MTS); SPADE Agent Communication Channel (ACC); Extensible Messaging and Presence Protocol (XMPP); SSL for security</td>
</tr>
<tr>
<td>Tryllian’s ADK</td>
<td>N/A</td>
</tr>
<tr>
<td>Use</td>
<td>Agent-Based Scheduling, Workstation Management, SMS Dating (Frog)</td>
</tr>
<tr>
<td>Misc.</td>
<td>based on a P2P library called JXTA; Transport Layer Security (TLS v. 1.0), cipher suites, X.509 certification for authentication</td>
</tr>
<tr>
<td>Voyager</td>
<td>N/A</td>
</tr>
<tr>
<td>Use</td>
<td>Universities Research Projects</td>
</tr>
<tr>
<td>Misc.</td>
<td>provides simpler and better Service Oriented Architecture(SOA); location transparency through forwarding chains of proxies; unique identifiers for agents; own security manager</td>
</tr>
</tbody>
</table>

This analysis reveals that: there are few MAPs which supports BDI model of agency such as 3APL, ADK, AgentBuilder, C-BDI, Jack and Jason and out of these 3APL and Jason are only open source products; PMADE has the provision of effective itinerary management; Tryllian’s ADK claims to provide ‘strong’ mobility while others mostly provide ‘weak’ mobility; majority of the platforms are academic research projects developed at the universities with sanctioned grants and are not maintained further; Security is still an issue as most of the java based systems rely on the basic security mechanism of JVM; ABLE, AgentBuilder, C-BDI, CybelePro, FIPA-OS, Jack, JADE, Tryllian’s ADK and Voyager are the few commercial products and out of these JADE is the choice of most of the researchers today due to java based, open source, FIPA compliance, effective communication mechanism, GUI and well maintained documentation; Agent Oriented Software(AOS Group) is the world leader in autonomous systems software with Jack and C-BDI feathers in his cap and has a hold on the real application domains of defense, oil and gas production, unmanned air and under water vehicles.

Since the MAP market is evolving very fast, we would like to insist on the fact that the platforms analyzed in this paper are evaluated given their current state at the time of writing this paper (Jan, 2014), and that their specifications may change as time goes on. In addition to these traditional heavyweight MAPs, there are other lightweight MAPs specifically designed for the resource-constrained Wireless Sensor Network (WSN) such as Agilla [79] and ActorNet [80] for Mica2/TinyOS, MAPS [81], AFME [82] and MASPOT [83] for Sun SPOT/Squawk Java ME Virtual Machine.

### 9. RELATED WORK

In the last few years, many researchers have focused on testing the performance of existing MAPs and such comparisons provide an overview of the different alternatives. Most of these studies highlight a lack of performance, security, communication, and scalability.
performance of JADE, Tryllian’s ADK and Safeguard Agent Platform (SAP). They claim that JADE performs better than the others because is built on Java RMI but give no proofs confirming this claim. Furthermore, other services apart from the messaging service should also be tested.

Mawlud-Yunus et al. (2004a, 2004b)[92][93] studied the performance behavior of Aglets and TACOMA in a distributed search using single and multiple agents. The results indicate that the two mobile agent platforms have similar behavior but their performance varies with underlying implementation, and the multiple agents approach performs better than the single agent approach in large networks. Su and Li (2005)[94] compared the performance of Prolog based IMAGO with Aglets, D’Agents and Kaariboga. IMAGO performs worse than the other three systems in migration and communication and performs better in agent’s creation only. Raquel et al. (2007)[95] compared Aglets, Voyager, Grasshopper, Tryllian, JADE, Tracy, and SPRING5 qualitatively and evaluate their performance in a variety of settings with an extensive set of experiments. Leszczyna (2008)[96] evaluate the publicly available platform implementations enlisted at FIPA site (http://www.fipa.org/resources/livesystems.html) with the criteria of platform maintenance and popularity. Author found that the continuously maintained platforms are ADK, JACK and JADE and out of these JADE is the only one which is well supported (documentation, mailing list, platform updates) and free.

Alberola et al. (2010)[22] performed an in-depth study, specifically on the messaging and directory services, of JADE, MadKit and AgentScape in order to find out to what extent the internal design of a MAP influences its performance. The JADE messaging service performs better than the messaging service of the other two MAPs. In MadKit, directory service is provided by the MAP, in a distributed way. In JADE and in AgentScape this service is centralized. The authors conclude that services like directory service, which stores information and receives a lot of requests, should be distributed among the various hosts in the MAP whenever possible, for improving the response time of these services.

In general, the different works where some MAPs are compared are partially outdated today, as new platforms and versions have appeared since the publication of such works.

10. CONCLUSIONS
The qualitative strengths of intelligent autonomous MAS make intelligent software agent paradigm an attractive choice for a wider and wider range of distributed applications. Researchers interested in this field should now focus more on the designing and practical implementation of MAS in various domains by using the existing MAPs instead of developing a new MAP. With this study, we expect to contribute to cover the need of an updated review of MAPs and also to encourage future work in the intelligent software agent technology.

11. ACKNOWLEDGMENTS
We sincerely acknowledge all the authors and commercial organizations listed in the reference whose work and products are analyzed and views are provided in this study.

12. REFERENCES


