IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **Volume 11, Sp.Iss 7 , 2022**

Addressing Mobility Challenges with Software Agents

V.PremaLatha[1], Dr.Nikhat Parveen[2]

[1] Department of C.S.E,Koneru Lakshmaiah Education Foundation (KLEF), Deemed to be University, Vaddeswaram, Green fields, Guntur, Andhra Pradesh, India-522302..

[2] Department of C.S.E,Koneru Lakshmaiah Education Foundation (KLEF), Deemed to be University, Vaddeswaram, Green fields, Guntur, Andhra Pradesh, India-522302..

DOI : 10.48047/IJFANS/11/S7/011

Abstract

"We provide an overview and comparative analysis of mobile agent systems in the rapidly evolving field of software agents. This overview briefly delves into the concept of mobility within the context of mobile code languages and explores its relationship with distributed computing, such as the client-server model, while also examining potential application areas. Additionally, we discuss the necessity of integrating mobility with other features."

"**Keywords:** agents, software agents, mobile agents, distributed computing, client-server model, artificial intelligence, distributed artificial intelligence, KQML."

Introduction:

Various interpretations of software agents have been employed across different subfields within the growing agent research community, often with distinct and disconnected contexts. Mobility is among the potential attributes attributed to an agent. Mobility denotes an agent's capacity to traverse a network, a concept primarily harnessed in the realm of distributed computing. While mobility is not an obligatory trait for the general realization of an agent, it presents numerous practical advantages in the establishment of agent-based computing environments.

The utilization of mobile agents for distributed computing has gained prominence in recent years. Within this domain, agent-based computing has been regarded as an extension of remote script program execution, with a particular focus on enhancing security. The objective is to evolve classical client-server computing into mobile agent-based computing. In this agent-based framework, executable programs are transmitted across the network instead of mere data. To expand on this concept, mobile agents can be envisioned as autonomous entities that traverse between agent servers or Agent Execution Environments (AEEs) to fulfill their users' objectives. The structure of this paper is as follows: In Section 2, we begin with a concise overview of the dynamic field of software agents. Section 3 delves into the key shared concerns related to the concepts employed in current mobile agent systems. Finally, in Section 4, we briefly explore the potential integration of mobility with other aspects of agent technology.

Software Agent

What characterizes an agent, and how does it differentiate from a "computer program," a "network node," or an "actor"? Defining the term "agent" has proven to be as challenging as defining "intelligence." Let's compare four distinct definitions of an agent:

"An agent is anything capable of perceiving its environment through sensors and taking actions within that environment through effectors." — AI agents (Russell and Norvig).
"An entity is considered an agent if it effectively communicates using an Agent Communication Language (ACL)." — Heterogeneous agents (Genesereth and Ketchpel).

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **© 2012 IJFANS. All Rights Reserved**, **Volume 11, Sp.Iss 7 , 2022**

"An agent is a computer program employing AI techniques to provide user assistance in a specific application." — Interface agents (Maes).
"An agent is a process that can migrate across a computer network to fulfill requests from its clients." — Mobile agents (Johansen et al).

Evidently, there is no universal consensus across different fields regarding the metaphorical use of the term "agent." Its meaning is contingent on the context in which it is applied. The subsequent three papers address the challenge of agent definition, each offering distinct classification approaches in their attempts to resolve this issue.

Wooldridge and Jennings categorized agent concepts into two distinct categories: weak and strong notions. The weak notion of agency encompasses the following components:
Autonomy: A system is considered autonomous to the extent that its behavior is influenced by its own experiences.
Social ability: This involves interacting with other agents, and potentially with humans, using an Agent Communication Language (ACL).
Reactivity: The ability to promptly respond to environmental changes.
Pro-activeness: Demonstrating goal-oriented behavior by taking the initiative.Conversely, the strong notion of agency pertains specifically to agents within the context of artificial intelligence (AI).

Beyond the weak notions, agents can also possess mentalistic concepts. By attributing mental states to agents, they can be viewed as intentional systems. Intentional systems encompass two significant categories of attributes:

- Information attitudes, which provide the agent with insights about the world, such as knowledge and belief.
- Pro-attitudes, which steer the agent's actions, including desires, intentions, obligations, and commitments.

An agent should possess at least one attribute from each of these categories. Additionally, other notions, such as mobility, rationality, believability, and more, can also be considered.

Franklin and Graesser propose a natural kinds taxonomy for autonomous agents, and they use a collection of agent features for further classification. After examining eleven distinct agency definitions, they suggest the following definition: "An agent is a system situated within and integrated with an environment that perceives and takes actions over time, pursuing its own objectives while influencing what it perceives in the future."

In the same year, Nwana classifies agents based on a typology. He contends that the term "agent" is not owned by any specific research community in the manner that "fuzzy logic," for instance, is associated with logicians and AI researchers. Instead, the term "agent" has evolved into an overarching descriptor encompassing a diverse array of research and development efforts. These classifications and definitions capture certain common aspects of an agent, yet they do not offer a unified definition. The primary challenges in achieving a unified definition are as follows: Numerous meaningful choices are available, some of which may inherently contradict one another.

A unified definition could limit the versatility of the agent metaphor.Over time, the metaphor may become increasingly ambiguous(Burkhard).In the subsequent subsections, we briefly provide an overview of a set of software agents that have been explored and developed in various fields (refer to Figure 1). Dashed lines in the figure indicate that agent research in certain domains mutually influences each other. This classification pertains to the field in which these agents are studied. Please note that this classification is neither rigid, as there can be overlaps in

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **© 2012 IJFANS. All Rights Reserved**, Volume 11, Sp.Iss 7 , 2022

agent functionality and concepts, nor comprehensive, as there are undoubtedly other types of agents (e.g., Interface Agents, Information Agents, and so forth) not covered here. The primary focus of this paper, mobile agents, will be discussed in a separate section.

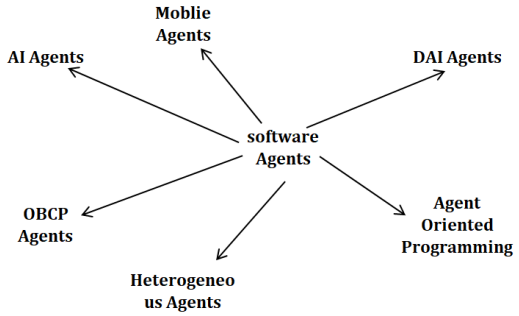

Fig. I. The paper delves into software agents. Dashed lines in the diagram signify the interplay between various fields, mutually driving the advancement of agent development.

2.1. Agents in AL

The concept of intelligent computer agents finds its origins in the early history of artificial intelligence (AI). These agents are not only the original inspiration but also the ultimate objective of AI research, as noted by Hayes-Roth. Surprisingly, the issue of synthesizing agents has received relatively little attention until just a decade ago. Given the formidable challenge of constructing an intelligent computer agent, the AI community adopted a "divide and conquer" strategy. Different AI subfields focused on various aspects of the agent's makeup, often without a clear vision of the ultimate goal—namely, the integration of these disparate methods and concepts into a comprehensive agent. The divergent development of these agent components created difficulties in their eventual integration, which remains a significant challenge in the field of AI.

In a recent AI book by Russell and Norvig ,the authors endeavor to contextualize nearly all AI research within the framework of constructing an AI agent. Figure 2 provides a simplified representation of an AI agent, which can be defined as any system capable of perceiving its environment through sensors and taking actions within that environment through effectors, as outlined in Russell and Norvig's definition. This viewpoint holds even in a software environment, making it applicable to a software agent situated in a software environment, such as the Internet.

Architectures: Various architectures have been explored for constructing the cognitive component of an individual agent. In the context of AI agents, three notable types can be highlighted:

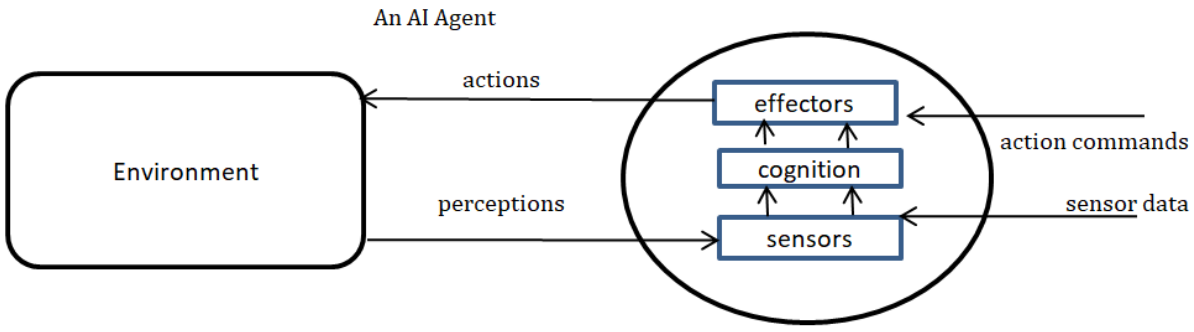

Fig. 2.An elementary perspective on an AI agent.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **© 2012 IJFANS. All Rights Reserved**, **Volume 11, Sp.Iss 7 , 2022**

Three fundamental agent architectures can be distinguished:

Deliberative Agents: These agents incorporate an explicitly represented symbolic model of the world where decisions are reached through logical reasoning. They assume that cognitive functionality can be compartmentalized. Deliberative agents possess the capability to reason about new situations and the behavior of other agents. However, their limitations include the computational complexity of general symbol manipulation and relatively slower response times.

Reactive Agents: Reactive agents are founded on a collection of simple behavior patterns, essentially stimulus-response behaviors that react to environmental changes. These agents continuously rely on their sensors without maintaining an internal symbolic model of the world. Architectures like subsumption (Brooks) and situated automata (Rosenschein) fall within this category. The challenges associated with reactive agents include their hard-wired behavior and their limited ability to learn, adapt, and predict.

Hybrid Agents: Hybrid agents leverage both deliberative and reactive architectures, combining high-level reasoning with low-level reactive capabilities. The reactive components handle time-critical actions, while the deliberative part guides the behavior of the reactive section, for instance, by modifying the set of situation rules governing the reactive part.

These architectural concepts pertain to individual agents. However, when dealing with a group of agents, additional factors must be considered when constructing a multi-agent environment. The examination of how a collective of interacting agents behaves falls within the domain of Distributed Artificial Intelligence (DAI).

2.2. Agents in DAI.

In the realm of Distributed Artificial Intelligence (DAI), the primary focus is on the study of systems where multiple agents engage in interactions. DAI can be divided into two key components: Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS). DPS primarily deals with the allocation of tasks aimed at achieving a common goal, involving the assignment of global tasks to a cooperative set of agents, each with the objective of maximizing a global utility function. On the other hand, MAS pertains to a group of autonomous agents, each possessing its own knowledge, local objectives, and capabilities, all within a shared environment. In this scenario, agents concentrate on solving local tasks and maximizing their local utility functions. While global situations can arise, agents may harbor competitive goals rather than shared ones. Transitioning from a single-agent scenario to a society of agents introduces a new set of considerations, including the critical aspects of coordination, communication, cooperation, negotiation, as well as theories of intention and action

2.3. Agents in Object-Based Concurrent Programming:

The concept of an actor, as introduced in the concurrent actor model, aligns with the idea of an agent. An actor is defined as "a computational entity with a mail address and behavior, capable of concurrent message passing and concurrent execution of actions" (Hewitt). In this context, an agent can be seen as an extension of an actor, which, in turn, is a self-contained, interactive, and concurrently executing object. Object-Based Concurrent Programming (OBPC) serves as a precursor to agent languages. OBPC is built on the concept of active and autonomous objects that perform concurrent computations and interact through message exchange. Given the considerations in Distributed Artificial Intelligence (DAI), as discussed earlier, there is a compelling case for extending OBPC to address DAI-related issues (Gasser and Briot).

2.4. Heterogeneous Agents: Heterogeneous agents refer to agents within the context of agent-based software engineering (Genesereth and Ketchpel). The primary motivation behind these agents stems from the growing demand for applications working in isolation to become

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **© 2012 IJFANS. All Rights Reserved**, **Volume 11, Sp.Iss 7 , 2022**

interoperable. Heterogeneous agents facilitate effective communication and knowledge exchange between applications in a networked computing environment.

An entity qualifies as an agent when it effectively communicates within the parameters of an Agent Communication Language (ACL). The architectural intricacies of a heterogeneous agent system are examined by the Knowledge Sharing Effort (KSE), with three distinct groups focusing on the achievement of interoperability among such agents. Agents are considered heterogeneous when their implementation programming languages, knowledge representations, and inference systems vary. To actualize the concept of an agent, KSE outlines the following: Knowledge Query and Manipulation Language (KQML) as a universal Agent Communication Language (ACL). KQML establishes a high-level protocol for agent interaction and communication, rooted in speech act theory.

Ontolingua serves as a set of tools for vocabulary maintenance, ensuring clarity when agents with differing vocabularies exchange knowledge.Knowledge Interchange Format (KIF) is designated as the standard knowledge representation language for facilitating knowledge exchange between agents.

2.5. Agent-Oriented Programming: Shoham introduces the concept of Agent-Oriented Programming (AOP) and, along with AGENTO, provides the first prototype of an AOP system. Notable distinctions between an object and an agent include the following: an object can possess arbitrary attributes, whereas an agent's attributes constitute a fixed set that defines the agent's mental state. Moreover, inter-agent communication employs a predefined pattern and is restricted to a set of messages akin to KQML, in contrast to inter-object communication, which carries fewer constraints. An agent perceives incoming messages, updates its mental state, and deduces the actions (e.g., outgoing messages) to be undertaken. Subsequently, new AOP languages rooted in AGENTO or the AOP paradigm have emerged.

Mobile Agents: In the context of mobile agents, mobility implies the capacity to relocate themselves to another location while maintaining internal integrity and continuing their execution.

3.1. Mobile Code Languages: Code mobility is facilitated through various means. Mobile code languages can be categorized into those supporting strong or weak mobility, as delineated by Carzaniga et al. Strong mobility entails the migration of both the code and the execution state of a program to a new host. Program execution is paused, the program is transmitted to the destination host, and execution resumes there. Weak mobility, on the other hand, involves the dynamic binding of a program to code originating from a different source, achieved by downloading from the network (as exemplified in Java by Sun) or receiving it from another remote program. Notably, Telescript (White) and Agent Tcl (Gray) fall into the strong mobility category, while Java facilitates weak code mobility through its class loader for network-based class downloads, and TACOMA (Johansen) allows the transmission of both code and data for remote execution.

3.2. Mobile Code Paradigms: Code mobility serves diverse implementations of distributed applications (see Figure 3). The prevalent client-server (CS) paradigm is commonly employed for distributed application realization today. Within the CS model, code mobility is generally unnecessary.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **© 2012 IJFANS. All Rights Reserved,** Volume 11, Sp.Iss 7 , 2022

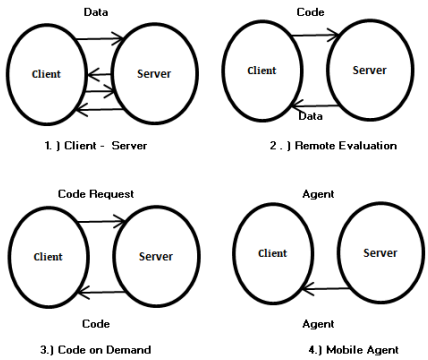

FIG 3Four approaches to distributed computing.

Client-Server:

In the client-server framework, a server offers a predetermined and non-configurable range of services. Clients access these services through RPC-based communication. All the necessary code, resources, and processing capabilities are situated on the server side.

Mobile Agent Systems:

The mobile agent model we've discussed can be expanded into what we'll refer to as mobile agent systems. Mobile agent systems represent a broader environment for agent execution. In these systems, agents operate within Agent Execution Environments (AEEs) throughout the network. They have the capability to migrate as needed to collaborate with other agents. Mobile agent systems not only provide an alternative to more traditional approaches (e.g., client-server) but also offer practical benefits in supporting limited local resources, asynchronous computing, a more intuitive development environment (e.g., for creating electronic marketplaces), and a flexible distributed computing architecture.

Today, there are environments available for implementing mobile agent systems, including both commercial solutions (e.g., Telescript by White and Aglet by Lange and Chang in) and research prototypes (e.g., TACOMA by Johansen et al. and MOLE by Strasser et al. in). Many of these systems are built upon shared concepts. This discussion explores the concepts and design space for a hypothetical mobile agent system—a distributed system that enables and supports the execution of mobile agents. This hypothetical system is informed by an analysis of current approaches to realizing mobile agent systems and the common concepts shared among these systems and the mobile agents that run on them.

3.3.1. The Mobile Agent: Mobile agents can be constructed using procedural components or through classes and objects. It's preferable to implement agents using interpreted languages, despite the performance trade-off, as they offer several advantages compared to machine languages. Interpretable code can seamlessly function on diverse platforms, accommodate references that may only exist at the agent's destination (late binding), and provide easier security management, as language developers can explicitly control access to critical system resources. A mobile agent typically comprises:

- The program code (program state).
- The content of instance variables (data state).
- The stack (execution state).

A basic mobile agent might consist of program code in the form of a script, with data state represented as assignments within the program. For efficiency, portions of code can also be dynamically fetched as needed.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper **Volume 11, Sp.Iss 7 , 2022**

Mobility: When an agent makes the decision to move, two possible scenarios unfold. Agent A can generate another agent, B, to send to a remote host. After performing its computations on the remote host, B will either return with the final results, during which time A can either suspend its execution or continue executing at the local host. Alternatively, agent A can halt its execution at a particular point, migrate to another host, and then resume execution from the point where it paused, much like the concept of process migration in an operating system (OS). Typically, the language provides a command such as "jump," "go," "move," or "migrate." These commands have varying semantics in different languages but all result in the program relocating to a remote host.

Autonomy: In the context of mobile agents, autonomy signifies that the agent's decision to migrate is a self-initiated choice and not imposed by an external entity, such as the OS. Processes that migrate under external influence, for the purpose of load balancing or enhancing parallelism, cannot be classified as mobile agents because their migration is compelled rather than self-determined.

3.3.2. The Agent Execution Environment: Within a mobile agent system, a network of distributed Agent Execution Environments (AEEs) is established. These AEEs can be dynamically created through runtime invocation. AEEs serve as computational environments interconnected by a communication medium, facilitating the execution of mobile agents. They are also referred to as Agent Meeting Points (AMP) in some contexts, as they serve as common locations for agents to convene and interact. Each AEE comprises at least one execution engine.
Security is a vital prerequisite for a mobile agent system because mobile agents must operate within a secure and trustworthy execution environment. Two key aspects of security issues in a mobile agent system are AEE security and the privacy and integrity of the agents.

Conclusion:

Towards the end of this section, we explore several application areas for mobile agent systems, including:

Network management: Mobile agents, acting as management scripts, enable the distribution of management activities across time and space in the domain of distributed network management. While most of these applications can be handled by stationary programs and traditional techniques like the RPC method, mobile agents offer improved performance and introduce a real-life metaphor, providing a more flexible framework for establishing an agent-based computing environment where legacy systems, stationary and mobile agents can collaborate.

At the technical level, mobile agents may not have strong conceptual ties to agents in other fields. However, when discussing technology applications, common features and connections with agents in various domains become evident. Mobile agents often operate in open distributed environments, such as the Internet, which has evolved into a vast and interconnected ecosystem. In this dynamic and uncertain environment, mobile agents must adapt, reason about unpredictable scenarios, and possess self-awareness and adaptability to maintain their required level of intelligence.

Introducing reflective behavior, as proposed by Tyugu and Addibpour, can enhance the capabilities of mobile agents. Reflective agents have been studied in both Distributed Artificial Intelligence (DAI) and Object-Based Concurrent Programming (OBCP). Furthermore, mobile agents should be capable of migrating across heterogeneous boundaries in distributed systems, which suggests the need for integrating mobile and heterogeneous agents.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **Volume 11, Sp.Iss 7 , 2022**

Reference

1. Adenaw, L.; Lienkamp, M. Multi-Criteria, Co-Evolutionary Charging Behavior: An Agent-Based Simulation of Urban Electromobility. *World Electr. Veh. J.* **2021**,
2. Lemiec, M.; Malinowski, K.; Szymoński, M.; Ganzha, M.; Paprzycki, M. Agent-based modelling of car platooning for traffic optimization. In Proceedings of the 2021 4th International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR), Batu Pahat, Malaysia, 6–8 September 2021;
3. Delhoum, Y.; Belaroussi, R.; Dupin, F.; Zargayouna, M. Multi-Agent Activity-Based Simulation of a Future Neighborhood. In Agents and Multi-Agent Systems: Technologies and Applications 2021
4. UCI Machine Learning Repository: Heart Disease Data Set. 2019.https://archive.ics.uci.edu/ml/datasets/Heart+
5. Disease[Internet]. Archive.ics.uci.edu. [cited 17 December 2019]. Available from: [Google Scholar]
6. Seven Tests to Diagnose Heart Diseases- the Times of India. 2020.https://timesofindia.indiatimes.com/7tests-to-diagnose-heart-diseases/listshow/43215326.cms[Internet].
7. Itagaki, M.W. Using 3D printed models for planning and guidance during endovascular intervention: A technical advance. Diagn. Interv. Radiol. 2015, 21, 338–341.
8. Zein, N.N.; Hanouneh, I.A.; Bishop, P.D.; Samaan, M.; Eghtesad, B.; Quintini, C.; Miller, C.; Yerian, L.; Klatte, R. Three-dimensional print of a liver for preoperative planning in living donor liver transplantation. Liver Transpl. 2013, 19, 1304–1310.
9. Chandak, P.; Byrne, N.; Coleman, A.; Karunanithy, N.; Carmichael, J.; Marks, S.D.; Stojanovic, J.; Kessaris, N.; Mamode, N. Patient-specific 3D Printing: A Novel Technique for Complex Pediatric Renal Transplantation. Ann. Surg. 2019, 269, e18–e23.
10. Silberstein, J.L.; Maddox, M.M.; Dorsey, P.; Feibus, A.; Thomas, R.; Lee, B.R. Physical models of renal malignancies using standard cross-sectional imaging and 3-dimensional printers: A pilot study. Urology 2014, 84, 268–272.
11. Murphy, S.V.; De Coppi, P.; Atala, A. Opportunities and challenges of translational 3D bioprinting. Nat. Biomed. Eng. 2020, 4, 370–380.