# Engineering Multi-Agent Systems

**Tagelsir Mohamed Gasmelseid**
*King Faisal University, Kingdom of Saudi Arabia*

## INTRODUCTION

The migration of business enterprises to decentralized operations, location independence, and micromanagement has been accompanied by the emergence of different computing paradigms, enterprise architectures, and communication platforms. Software agents perform some tasks on behalf of their users, other agents, or programs with some degree of autonomy using multiple information and communication platforms. The use of wireless devices and networks has significantly improved information transmission and transaction processing in support of virtual and physical mobility and the acquisition, customization, and use of context-specific information for electronic and mobile shopping, finance, banking, and payment services.

## BACKGROUND

The proliferation of networked and Web-based information systems shows a growing interest in using multi-agent systems in different applications (electronic commerce, airlines, insurance, distance learning, manufacturing, and the management of common pool resources) because of their potential decision support, negotiation, and task-delegation features. Software agents (making up multi-agent systems) perform some tasks on behalf of their users. These tasks range from information search and retrieval, management of information overload, scheduling and interface presentation, task delegation, user training, event monitoring, and information search, to matchmaking and decision making. Multi-agent systems offer a new dimension for coordination and negotiation by incorporating autonomous agents into the problem-solving process and improving coordination of different functional unit-defined tasks, independent of both the user and the functional units under control (Byung & Sadeh, 2004). Their capacity to carry out these tasks demands that they possess some basic qualities including autonomy, conviviality, reactivity, learning, mobility, benevolence, rationality, and adaptivity (Lai & Yang, 2000; Jung & Jo, 2002; Lisa, Hogg, & Jennings, 2001; Hu & Weliman, 2001). The growing use of multi-agent systems in different domains has also been accompanied by an expanding interest in "mobility," "context awareness," and "information security."

Mobility allows different agents to move across different networks and perform tasks on behalf of their users or other agents, by accessing databases and updating files in a way that respects the dynamics of the processing environment and intervention mechanisms. The importance of maintaining security in mobile systems stems from the importance of maintaining integrity, privacy, and information sharing. Furthermore, context awareness allows multi-agent systems to support mobility through the acquisition and use of context information that describes location, time, activities, and the preferences of each entity. The dynamic and adaptive provisioning of context information requires an expressive, semantically rich representation to support context information acquisition, context engagement (required by certain events to trigger actions), and context dependency (the relationship between different aspects of context information). On the other hand, seamless concepts require a service layer in the multi-agent infrastructure that is capable of delivering functionalities such as context management, context-based service discovery, and a communication protocol responsible for handling issues such as presence, notification, and privacy (Khedr & Karmouch, 2005).

Within this context, emphasis on "information security" continued to be made on technological solutions and the use of hardware devices or computer programs. The basic aim is to prevent, avoid, detect, or prepare for breaches of security that threaten the confidentiality, integrity, or availability of information processed by computer systems. The majority of networked systems are managing their information security through the use of cryptographic algorithms, digital signatures and challenge response authentication techniques, hash algorithms, and hybrid encryption mechanisms and protocols (Microsoft, 2000). Asymmetric cryptographic algorithms use two related keys (public and private), each of which has the characteristic algorithm that, given the public key, is computationally infeasible to derive the private key. Symmetric cryptography, on the other hand, transforms (encrypts) the plaintext (original data) into ciphertext (protected data) in a way that makes it infeasible to reverse the process without the full knowledge of the transformation function. A hash function is a one-way transformation that efficiently turns arbitrary-length data into fixed-length data, and gives some data or its hash value. However, it is computationally infeasible to find some other data that will hash into the same value. Hash algorithms are commonly used for digital signatures, passphrases, integrity protection, and challenge-response authentication. Applications frequently employ hybrid or bulk encryption when they are required to apply a confidentiality service to shared data. Using a protocol such as (SSL/TLS) processing is done on the assumption that the receiver has a private-public key pair and that the sender has obtained the public key. Using hybrid encryption and hash functions, digital signatures offer a data authentication service and ensure the origination of messages from the source and stability of contents. However, approaching information security through technological solutions is challenged by the variety of key length, computational complexity, and breach possibilities. Therefore, it is essential to "couple" technological solutions with an array of other factors (human resources, standard operating procedures, structure, and system development methodologies) that should be investigated when addressing information security.

## INFORMATION SECURITY OF MULTI-AGENT SYSTEMS: REVISITED

The process of developing multi-agent systems continued to be guided by different agent-oriented software engineering (AOSE) methodologies such as Gaia, Tropos, MESSAGE, Prometheus, and MaSE. While some current AOSE methodologies are "expanding" the application of existing "conventional" object-oriented methodologies to agent-oriented domains, others are focusing on defining a number of models that guide the process of designing agent-oriented applications in accordance with the basic guidelines of agent theory (Wooldridge, Jennings, & Kinny, 2000). Some of these methodologies are criticized for their limited deployment due to the lack of maturity (Dam & Winikoff, 2003) and their failure to capture the autonomous and proactive behavior of agents, as well as the richness of the interactions (Zambonelli, Jennings, Omicini, & Wooldridge, 2001). Current agent-oriented methodologies focus mainly on multi-agent systems analysis and design, but without providing straightforward connections to the implementation of such systems (Mercedes et al., 2005). They are characterized with a fundamental mismatch between the concepts used by object-oriented developers and the agent-oriented view (Wooldridge & Jennings, 1999). As a result, they fail to adequately capture an agent's flexible, autonomous problem-solving behavior, the richness of an agent's interactions, and the complexity of an agent system's organizational structure. Most of these methods feature a technology-driven, model-oriented and sequential approach, and assume (in advance) the suitability of multi-agent technology for the development of multi-agent applications, which may not always be the case in different problem domains. Because model orientations of these methodologies are obvious, the process of model coupling and integration does not explicitly reflect the links between models (Lind, 1999). Besides the main issues (known as agent qualities) to be addressed by agent-oriented software engineering methodologies (such as autonomy, reactivity, proactiveness, and social ability), the concern for mobility has been growing over time (Pablo et al., 2003).

In spite of the growing diffusion of mobile agent technology, little research has been done to settle "design" directions to be followed in order to determine when mobile agents are convenient to be used or not. However, the current agent-oriented software engineering methodologies used for developing multi-agent systems do not provide methods to determine in which cases mobile agents should be used. Many of the existing methodologies intentionally do not support intelligent agents; rather, they aim for generality and treat agents as black boxes (Padgham & Winikoff, 2002).

While the entire agent-oriented software engineering methodologies have provided alternative ways for describing "tasks" and "relationships," little has been done to incorporate "information security" considerations in multi-agent "mobile" and "context aware" applications. The importance of maintaining the in-