Advances in Fault-Tolerant Multi-Agent Systems

Lúcio Sanchez Passos
LIACC, University of Porto, Portugal

Rosaldo J. F. Rossetti
LIACC, University of Porto, Portugal

Joaquim Gabriel
LIACC, University of Porto, Portugal

INTRODUCTION

Multi-Agent Systems (MAS) arose in the early 1980's as a promising software paradigm for complex distributed systems. It derives from an Artificial Intelligence subfield concerned with concurrency of multiple intelligent problem-solvers, known as *Distributed Artificial Intelligence* (DAI). According to Gasser (1987), MAS "is concerned with coordinated intelligent behavior among a set of (possibly pre-existing) autonomous intelligent 'agents:' how they can coordinate their knowledge, goals, skills, and plans jointly to take action or solve (possibly multiple and independent) problems." Since then, the concept of autonomous agent has been broadly studied in diverse fields.

In spite of the vast number of research achievements in a couple of decades and successful industrial applications such as ARCHON (Wittig, 1992), the body of knowledge on MAS has not yet experienced the growth of practical implementation in real complex distributed systems. Several works (McKean, Shorter, Luck, McBurney, & Willmott, 2008; Pěchouček & Marík, 2008) have pointed to the under-exploration of the multi-agent approach in industrial environments, and thus this issue motivates the foundation of the Technical Forum Group (TFG).

Seeking for enlightenment to the question "Why not Multi-Agent Systems are largely used in real complex (distributed) systems?" the TFG wrote a document (Mcburney & Omicini, 2008) in which they pointed out several issues of the theme. One of them is the *risk* related to the implementation of this new approach which has never showed its value in large-scale problems. Such negative perspective comes from the industry's "fear" of the emergent behavior of MAS without any central decision unit (Pěchouček & Marík, 2008). This issue by itself makes the industry to avoid the use of MAS to control their critical tasks. To overcome this issue, it is necessary to expand the MAS capabilities in order to assure that the system is prepared to deal with unexpected situations on a sound and safe basis.

Hence a critical aspect emerges from this discussion: the *dependability* of software systems. Software dependability is a property that establishes a correlation between the reliance and services delivered by the system; *i.e.* the overall assurance depends on how an application behaves in the user(s)' perspective (be it human, hardware, or another software). As noted by (Laprie, 1995), dependability has multiple, but complementary, attributes, namely: *confidentiality*, *integrity*, *safety*, *availability*, *reliability*, and *maintainability*. Confidentiality and integrity are associated with the security of the software and seek to protect it against unauthorized access and modification to information, respectively. The remaining attributes share one point in common: they all seek to ensure reliance.

The *means* to achieve the desirable level of reliability must be an important point of discussion from the very beginning of a software construction, and then it should progress through the validation phase. According to Pullum (2001), these methods fall into four major groups: fault avoidance, fault removal, fault forecasting, and fault tolerance. As this work focus on runtime faults, *fault tolerance* techniques must be extensively understood because they aim to ensure complying services in the presence of system faults, *i.e.* when a

DOI: 10.4018/978-1-4666-5888-2.ch690

fault occurs. Those schemes provide mechanisms to avoid overall failure after the system deployment. Regarding the agents' context, such mechanisms also have been applied to improve their reliability aiming to achieve a *Fault-Tolerant MAS*.

BACKGROUND

Multi-Agent Systems can be seen as a computational system composed by various entities named *agents*. Each of them (agents) is able to autonomously reason and act upon a certain environment leading to the satisfaction of its own (and sometimes shared) goals. The definition of such an entity was one of the very first matters discussed by the community and, as result, diverse descriptions were proposed. Yet, none of them completely comprised all possible features of an agent; nonetheless properties such as *reactiveness*, *autonomy*, *goal-orientation*, *persistency*, *sociability*, *intelligence*, *robustness*, *reliability*, and *adaptability* are the most cited in several definitions (Weiss, 2000). Furthermore, the implementation of agents demands for an internal architecture and, therefore, it might be from reactive (composed by a simple set of rules) to deliberative (based on mental states) passing through hybrid (a mixture of both approaches).

All non-agent elements of a Multi-Agent Systems are typically considered to be part of the environment. In the beginning, the environment was seen as an implicit part of MAS being dealt with as an *ad-hoc* manner. Then it was promoted to a *per si* concept demanding careful attention. Hence the environment can play different roles in Multi-Agent Systems. It serves as a container and means provider for communication embracing interaction protocols. Regarding the social aspects of MAS, environment is seen both as an organizational layer and, generalizing (Weyns et al., 2007), as a social behavior infrastructure.

Merging all the pieces described above, a Multi-Agent Systems is a computational system in which a set of agents co-habit in an environment and interact in pursuing some set of goals. This set may include its own goal(s) and/or collective goal(s) (Watt, 1997). In a MAS each agent has limited and different capacity of perception and action upon the environment. That is, each agent has a distinct circle of influence being it just able to influence certain parts of the environment (Weiss, 2000). These circles of influence may overlap depending on agents' relationship and from this some social behaviors may arise. As such, agents negotiate coordination due to the capacity to act upon overlapped circles, or compete for a resource that might be important for achieving their goals.

Formally, MAS research concerns several aspects of agents, from the manner of how they coordinate their plans jointly to take actions or solve problems, to designing agents' reasoning processes and skills. Also, it focus on the development of principles and computational models to design, implement, and analyze the behavior and mutual interaction of agents when they are in societies in several dimensions.

THREATS IN MULTI-AGENT SYSTEMS

Before diving into studies of fault tolerance in MAS, there is a need to better understand what threats this kind of systems. The multi-agent metaphor is dual when looking at the fault handling facet. One point of view says that MAS are inherently dependable, that is, it perfectly conforms to one of the basic principles for building fault-tolerant systems: the modularity (Hägg, 1997). The agent is a self-contained entity which does not rely on external factors to exist (however, of course, it depends on external factors to interact and to fully execute its functions). On the other hand, multi-agent systems and fault tolerance diverge since there is no absolute mechanism either to synchronize or to control the former and it is intrinsically *non-deterministic*. This issue is related both to the programming nature of agents and to their autonomy.

Much research inherits definitions of threats given by previous authors in the field of fault tolerance and exception handling, such as (Avižienis, Laprie, & Randell, 2004) and (Liskov & Snyder, 1979) respectively, and adapt them to agent-based systems. For instance, Tripathi and Miller (2001) and Souchon, Dony, Urtado, and Vauttier (2004) see an exception as an event that is caused exclusively by a failure in the language constructor or by the underlying virtual machine; likewise Xu and Deters (2004) describe a failure as a joint of particular conditions and the presence of a fault. Still, these definitions do not completely comply with the agent paradigm due to the lack of reference to issues