Chapter 7
Exploring a Self Organizing Multi Agent Approach for Service Discovery

Hakima Mellah
Research Centre in Scientific and Technical Information - CERIST, Algeria

Soumya Banerjee
Birla Institute of Technology, International Center, Mauritius

Salima Hassas
University of Lyon, France

Habiba Drias
USTHB, Algeria

ABSTRACT
The chapter presents a Multi Agent System (MAS) approach, for service discovery process to consider the user in the service discovery process involving his interactions under constraints. The service discovery has become an emerging phenomena in software engineering and process engineering as well. The proposed MAS has demonstrated significant self organizing potential. This feature is very crucial for assuring a correct service delivery, to avoid failures or mal-function for the service discovery environment. The requirement for self organizing choreographed services have been well realized, in case of operational, functional and behavioral faults. Self organization within the MAS is adopted by the recourse to a self organizing protocol conceived from bacteria colony and evolutionary computation paradigm.

1. INTRODUCTION
Service orientation promotes a new way to design and implement large scale distributed applications across organizational and technical boundaries. However, it does not provide sufficient means to cope with the increasing complexity in service-oriented software applications. A promising way to get rid of the dilemma is to enable self-organization in service oriented computing (Gonzalez, 2005) (Martin, 2006). The emphasis could be given to the service orientation in the architectural design facilitates reusability, flexibility, interoperability, and agility.
for the software engineering part of those kind of system. Broadly, the service autonomy raises the question of how to establish proper operation status on the system level especially in presence of possible failures in some service elements. The research exhibit that the intelligent agent could be self organized to anticipate and follow up the failure part of software engineering. Practically, the proposed model has the pragmatic feature to trace he distributed problem and it is also capable to adopt the instant maintainability of the system. In this context, it is important to reciprocate the service of agents within the system and to get back the re-engineered system with minimum alterations. The agents are participating in these processes and re-discover the ambiguous part of the system itself by tracing the protocol exchanged through agents and the system.

The rest of the chapter is organized as follows: Section 2 describes background and related works of the Web Services which is one of the prominent examples of Multi Agent System. Section 3 elaborates the self organization principle within the Multi Agent System. Section 4 presents a brief outline on protocol and its primitives followed by the detailed service discovery approach in section 5. Section 6 is devoted about the different failure of system and section 7 discusses a case study presenting the applicability of the proposal. Section 8 presents the necessary tools being used to implement the system. Finally section 8 and 9 present the conclusion and further research in this direction.

2. BACKGROUND

Web Services (WS) reliability is strongly dependent on the fault handling mechanisms of the communication protocols and on the messaging infrastructure mediating their interactions (Erradi, Maheshwari, & Tosic, 2006). Correct service delivery continuity defines web services reliability. This implies zero, or at worst, relatively few failures and rapid recovery time (Erradi, Maheshwari, & Tosic, 2006). WS are managed by different geographically distributed providers. The unreliability of any of the constituent WS could lead to the failure or QoS degradation, even if other constituent seem to be reliable. A service management middleware is proposed in (Erradi, Maheshwari, & Tosic, 2006), it is based on some recovery policies (eg retry, skip, use). In (Ardissono L., Furnari, Goy, Petrone, & Segnan, 2006) a framework is proposed for WS orchestration, as environment of this latter could present some exceptions that are not able to identify precisely their causality. WS are used to diagnose exceptions in a more precise manner. In (Ardissono L., Furnari, Goy, Petrone, & Segnan, 2007) an interaction protocol for every cooperating web service $W_S_i$ is represented by an abstract process. The local view of $W_S_i$ on choreography is determined by associating its abstract process to the final WS. In this context, a WS monitor is responsible of checking the choreographed services by receiving their status messages during choreography. The problem is also posed when the WS monitor itself does not respond in time.

(Pat, Lyu, & Malek, 2006) proposed a general approach in system fault tolerance that can be applicable to WS. It is based on a replication driven WS system and on a replication manager. The replication manager keeps check the WS availability by the polling method. All the checking process is centralized on the replication manager and its abstract process. The whole system seems to be non-functioning in absence of replication manager. The several other contemporary proposals are published in supporting the development of reliable composite web services and in centralized WS orchestration (Ardissono L., Furnari, Goy, Petrone, & Segnan, 2006). In decentralized orchestration, the state of the composite WS depends on its distribution across nodes. To discover and utilize services, human based approaches, is not only time consuming, but also requires continuous user interaction. Software agents have been subject