WHAT, WHY AND HOW TO INTEGRATE
Self Organization within a SOA

Hakima Mellah
Research Center in Scientific and Technical Information, Cerist, Ben Aknoun, Algiers, Algeria

Salima Hassas
Liesp, Lyon 1 Univeristy, Lyon, France

Habiba Drias
USTHB Univeristy, Algiers, Algeria

Keywords: SOA, Self Organization, MAS, Agility.

Abstract: The main point of this work is to show how to contribute to information system (IS) agility by enabling self-organization of distributed IS that represent the nodes of information network relating these IS. The contribution is focused on presenting some factors that lead and trigger for self organization in a Service oriented Architecture (SOA) and propose how to integrate Self Organizing(SO) mechanism as this latter has already been proposed(in another work) for a Multi Agent System(MAS).

1 INTRODUCTION

Nowadays Companies are faced with an increasingly complex environment where requirements of speed and effectiveness determine their competitiveness. An agile enterprise is a company with an information system synchronized with its business (MIBS, 2006; Syntec, 2007; McCoy and Plummer, 2006). The resources in terms of services offered by its information system (IS) may extend beyond the borders of the business using transit interfaces such as Internet. These interfaces can be used to interact with a variety of companies (other organizations) to request their services. Inversely an organization providing any service is often called upon to interact with a group of services seekers (Benatallah et al., 2002; Mellah et al., 2006). However a service can be changed and may evolve over time (Macella and Pernici, 2006); a new version of a service can be used in place of the old one. During organizations interaction process, new ones may integrate into this latter and offer new services with advanced features. The main purpose of computing services is to create the foundation technology and management required to support the enterprise agility (Zhao et al., 2007). To ensure this agility and allow use of services, in order to meet the business process and the user, it is important to have a service-oriented software architecture (SOA) (Zhao et al., 2007; Syntec, 2007). During the interaction, the meaning carried by information is related to the information itself, to the manner it is used and its relationship to other information in the same context or the same information network. This is even more important when a range of distributed and interconnected information sources is considered. The complexity theory provides a conceptual working environment, a way of thinking and a way of seeing the world (Cachon et al., 1999; McKelvey, ; Mitleton-Kelly, 2003) or just the information. This new approach requires new connections and is to see a new function (Cachon et al., 1999)for a portion of an entity spatially located. We look at the organization as a complex evolving system, co-evolving (Cachon et al., 1999)with a social ecosystem. Complex systems are multidimensional systems and all dimensions interact and influence each other. Self-Organization (SO) is a key feature of these systems. In an organizational context it can be described by the spontaneous arrival of a group for executing a task or reaching a goal. The group decides by itself what to do, how and when, without any external entity decision (Mitleton-

Kelly, 2003). Emerging properties, patterns or structures appear from individual elements interactions. In (Mellah et al., 2008) a representation model of distributed information sources is proposed and the information is structured in a three dimensional system taking into account content, use and structure. The contents that characterize an information source are also used for one or more uses, themselves in relation with other uses and other contents that are related with a structure or an informational network. Despite the potential of SOA to align business and enterprise information technology, SOA still exhibits a set of characteristics that make it complicated to apply self-organization to an SOA-based system(Lei et al., 2008). In (Lei et al., 2008) a reference architecture is proposed to enable controlled self-organization in a service-oriented environment. The distributed and heterogeneous characteristics of service-oriented computing demand for comprehensive management approaches. In the mentioned architecture, self-organization of SOA elements is achieved by a separate management overlay observing and controlling the underlying SOA layer. It is also expected that the architecture should be controlled by external policies supplied by human system participants. This kind of self organization is criticized by Gasser as it is mentioned in (Mellah et al., 2006).

2 SELF ORGANIZATION WITHIN A SOA

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 applications. A promising way out of this dilemma is to enable self-organization in service oriented computing (Lei et al., 2008). Using service orientation in the architectural design facilitates reusability, flexibility, interoperability, and agility of this kind of systems. Generally, service autonomy raises the question of how to establish proper operation status on the system level(Lei et al., 2008), especially in presence of possible failures in some service elements. Self-organization means the process of generation, adaptation and change of organizational structure. The latter is the result of individual choices of a whole of entities to begin in interaction in certain organizational diagrams. We should not determine the behaviour of a complex system but rather, one has to expect new possibilities. Thus, we will be able to adapt when the unforeseen ones arrive, because we will be ready to expect unforeseen (Mellah et al., 2006). This definition applies also to the many facets of self organization called self-x properties (Schmeck, 2005). For example, a system is self-healing, if it can eliminate the effects of malfunctioning units without needing for this corrective action any external assistance. Obviously, this requires the capability to detect deviations from the correct functionality of the system on a global or local level(Schmeck, 2005). Nevertheless, in nature many examples exist related with robust systems, are observed as being able to provide certain functionality in a completely self organized way. Therefore, there is some hope that methods observed in natural systems might be transferable to technical systems. Among them we quote ant colonies and bacteria colonies. The latter has been adopted to propose a self organizing protocol based on multi agents system (Heylighen, 2003).

2.1 Choreography versus Web Services Organization

The W3C (W3C Glossary) define orchestrations as ”the model of interaction that must comply a Web service agent to achieve its goal”. However, even if orchestrations are a support for incremental programming in response to the introduction of a new event or the coordination with a new service, they do not support their own composition(Peltz, 2003) . While orchestration describes, in terms of service, interactions it may have with other services, and internal stages of data processing or invocations of internal modules(Peltz, 2003), choreography describes the collaboration between services collection, whose aim is achieving a given objective. The achievement of this objective is done by exchanging messages (Austin, 2004). Therefore, in choreography it is possible to have multiple orchestrations and in each one, one service will act as an orchestra chef. In (Mellah et al., 2006) organization functioning influences much it’s structuring; its key factors are the elements which form part of it: tasks and activities, competences and responsibilities, interactions network as well as the bonds, which connect these elements. Instead of considering information sources connection, we consider the whole of WS that are choreographed to respond to a user request. These WS translate existence of a discovery structure. As in an organization, an SOA basic elements (WS), constituting a discovery structure, are connected by varied and complex flows which are all significant. They explain how the discovery process has been achieved, or which services are implied and composed relatively to the user query. The complex flows represent the organizational structure and choreographed services are mapped to this organizational structure's elements. The interactions networks connect the organizational positions (geographical positions on which services are located). The information network must be self organized in this context.

3 WHY IS IT IMPORTANT TO INTEGRATE SELF ORGANIZATION IN A SOA?

In services discovery process, information retrieval includes two orthogonal components (Mellah et al., 2006) which must be complementary and that are:
- The construction of the network: is a self-organizing process which requires an adaptive behaviour while respecting the network consumption;
- Research: is ensured by a distributed algorithm which will exploit the structure of the emergent network.

In (Mellah et al., 2006; Heylighen, 2003) a trigger factor for self organizing distributed information sources is IS response time, which can avoid a failure production in the whole system. IS may encapsulate services interact for responding to a user query. The encapsulation will be based on a service discovery structure that needs to maintain its connectivity (between services). While interacting, and in order to avoid resources bottleneck or single points of failures, services components or resources should maintain connectivity with each others. This connectivity can be assured by the recourse to self organizing resources or services components. Services fault causalities are varied and three main categories are distinguished (Erradi, 2006): (i)Functional faults that are caused by a non completion of task execution or some incorrect results delivered by service (for example: price limits or delivery deadline),
(ii)Environmental faults that refer to communication infrastructure exceptions and middleware failures of the hosting servers and database servers (for example: service response time, service availability).
(iii)Contractual fault that are linked to the violation of service level agreement (SLAs) and collaboration policies; in this case the service execution might be completed without the conformity of results to the negotiated SLAs.

In this paper, we emphasize more on the second category, and delayed service response time or non service availability when:
-machines or network connections of a particular service provider are currently overloaded. The information may influence an agent to choose a different provider offers the same or a similar service.
-a transaction was involved by a user, such as the provider identity and network locations of specific services within a composed service, we should avoid situation like resource bottlenecks or single point of failure within the network.
The quoted points are few factors to explore about the discovery of the required service(s). These can be triggered elements, dealing exactly with service discovery self organization within a SOA.

4 HOW SELF ORGANIZATION CAN BE INTEGRATED IN A SOA

To be able to integrate self organization within an SOA, we first consider the mapping between choreographed services and the organizational structure's elements, and then we will try to adapt self organization within services itself. In (Chan et al., 2006), a general approach in system fault tolerance that can be applicable to WS is proposed, 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. If the latter does not respond any more what will happen to the whole system? In (Ardissono et al., 2007) a WS monitor is charged 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. Several other proposals are supporting the development of reliable composite web services, in centralized WS orchestration (Ardissono et al., 2006). In decentralized orchestration, the state of the composite WS depends on its distribution across nodes. Human based approaches, to discover and utilize services, is not only time consuming, but also requires continuous user interaction. Software agents have been subject to research in many interrelated fields. They are long-lived, having persistent computations that can perceive reason, act, and communicate. They have the ability to make decisions independently, without human intervention and without influence from other agents, notably when they are some failures during the service discovery process. Chafle (Ardissono et al., 2006)proposes a framework based on local monitoring agents for checking the state of the orchestrated WS, and interact with a Status Monitor. Palathingal and his colleagues (Palanthingal and Chandra, 2004) developed a multi agents approach for service discovery and utilization. In this approach there is no indication on the adopted interaction protocol, while agents interact, or on failure production within the system. The most concrete example of a self organization is the way in which the ants produce pheromone like traces between the food sources. Thus, the food sources are organized in effective network of provisioning ways. Marco Dorigo (Gershenson and Heylighen, 2003) is a pioneer in the field of ant algorithms. The Bacteria colonies also reorganize by forming patterns in order to fight against the adverse life conditions, of the environment. They develop a sophisticated co-operative behaviour and complex possibilities of communication, in order to be able to change their pattern and thus to reorganize, like the direct cell-cell physical interactions by the intermediary of additional membrane polymers, which is useful for the model formation in ”without-life” systems. Based on these models, we have elaborated a communication model (Mellah et al., 2006) and a self organizing protocol (Mellah et al., 2007) based principally on communication primitives that show how, through interactions (communication), interacting information sources are self organized. Moreover, the latter is regarded as an interaction cellular program and complexity is ensured by the composition of blocks (primitives). In (Mellah et al., 2008) despite the model, a layered architecture is proposed for discovering contents of distributed IS, and the first layer that interacts with users, is based on agents. For this, we suggest the recourse to a Multi Agent Systems (MAS) self organizing protocol for service discovery process. The self organizing protocol is inspired by bacteria colony life.

5 WHAT IS TO BE SELF ORGANIZE IN A SOA

In Palathingal’s (Palanthingal and Chandra, 2004) approach, the proposed MAS is within the UDDI. When a service provider expresses the need to publish services, an agent is created. Each software agent is specialized in a category of services. In the proposed approach solicited in this paper, the proposed MAS for services discovery is based principally on data duplication, which is inspired from (Palanthingal and Chandra, 2004; Erradi, 2006), within the services provider. Therefore, instead to affect a provider of services to a single machine,services are duplicated on several machines and at each one, a software agent is associated. At each provider will correspond several machines. When a service does not respond any more or there is a resource bottleneck, another machine will be ready to provide the same or similar service. Based on these postulates, services will be considered as the tasks transferred between agents machines. The latter’s interaction will be based on the self organizing protocol proposed in (Mellah et al., 2007), and failures are detected by the checking primitive that keeps alive interactions between agents machines (figure1). The positioning primitive states the distance between agent machines that do not responds to a service request and on which data are duplicated; the routing primitive elaborates the way to follow for reaching the required service from other machines. After finding the addresses, the new service will be updated within the WSDL document. It will be also interesting to extend the MAS approach, for contexts providers, and resolve the same failures problems encountered, additionally with contexts.

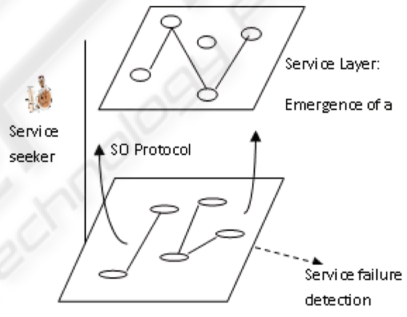

Figure 1: Emergence of a new services communication structure by the mean of the MAS AO protocol.

services are duplicated on several machines and at each one, a software agent is associated. At each provider will correspond several machines. When a service does not respond any more or there is a resource bottleneck, another machine will be ready to provide the same or similar service. Based on these postulates, services will be considered as the tasks transferred between agents machines. The latter’s interaction will be based on the self organizing protocol proposed in (Mellah et al., 2007), and failures are detected by the checking primitive that keeps alive interactions between agents machines (figure1). The positioning primitive states the distance between agent machines that do not responds to a service request and on which data are duplicated; the routing primitive elaborates the way to follow for reaching the required service from other machines. After finding the addresses, the new service will be updated within the WSDL document. It will be also interesting to ex-

tend the MAS approach, for contexts providers, and resolve the same failures problems encountered, additionally with contexts.

6 CONCLUSIONS

In this paper we have presented some challenging situations showing how and where to combine self organizing solutions with a SOA based systems, and what a MAS self organizing protocol can bring to a SOA. The proposed MAS self organizing protocol is inspired form bacteria colony. As perspective we are trying to assure services discovery process based on a self organizing MAS. The MAS will be integrated in a layered architecture combining agents and services technology.

REFERENCES

Ardissono, L., Furnari, R., Goy, A., Petrone, G., and Segnan, M. (2006). *Fault tolerant WS orchestration by means of diagnosis*. LNCS 4344, Springer Verlag, Berlin.

Ardissono, L., Furnari, R., Goy, A., Petrone, G., and Segnan, M. (2007). Monitoring choreographed services. In *Innovations and advanced techniques in computer and information sciences and engineering*. T. Sobth (Ed.).

Austin, D. P.-T. (2004). Web services choreography requirements w3c working draft. In *World Wide Web consortium (W3C)*.

Benatallah, B., Dumas, M., Fauvet, M., and Rabhi, F. (2002). Towards patterns of web services composition. In *Patterns and Skeletons for Parallel and Distributed Computing*. F.A. Rabhi and S. Gorlatch. Springer.

Cachon, G., Zipkin, P., and Anderson, P. (1999). Complexity theory and organization science. In *Organization science*, volume 10.

Chan, P., Lyu, M., and Malek, M. (2006). Making services fault tolerant. In *Third International Service Availability Symposium, ISAS 2006*, Finland. Springer verlag.

Erradi, A. (2006). Recovery policies for enhancing web services reliability. In *IEEE International Conference on Web Services, ICWS*.

Gershenson, C. and Heylighen, F. (2003). When can we call a system self organizing? In *In W. P. Banzahf, (Ed.), Advances in Artificial Life, 7th European Conference, ECAL2003*.

Heylighen, F. (2003). *The science of self organization and Adaptativity*. EOLSS Publishers Co. Ltd.

Lei, L., Thanheiser, S., and Schmeck, H. (2008). A reference architecture for self-organizing service-oriented computing. In *International conference on Architecture of Computing Systems ARCS.*, Germany.

Macella, P. and Pernici, M. (2006). Cooperative information systems based on a service oriented approach. In *Journal of Interoperability in Business Information Systems*. IBIS.

McCoy, D. and Plummer, D. (2006). *Defining cultivating and measuring enterprise agility*. Gartner Research, USA.

McKelvey, B. Complexity theory in organization science: Seizing the promise or becoming a fad? In *Emergence, journal of complexity issues in organizations and management*.

Mellah, H., Hassas, S., and Drias, H. (2006). Information systems self organization: Problematic, principles and methodologies. In *2nd International Advanced Database Conference- IADC*.

Mellah, H., Hassas, S., and Drias, H. (2007). Towards a self organizing protocol for a multi agents system (massop). In *the sixth IEEE international conference on Networking, ICN 07*.

Mellah, H., Hassas, S., and Drias, H. (2008). Modelling information in a three dimensional system for a contribution to the agility of an information system. In *Communication of SIWN*.

MIBS (2006). *Les infrastructures critiques:bien matriser le socle du patrimoine informatique de l'intreprise*. www.ib group.com/pageshtml/livre-blanc.

Mitleton-Kelly, E. (2003). *Complex systems and evolutionary perspectives on organisations: the application of complexity theory to organisations*. Elsevier.

Palanthingal, P. and Chandra, S. (2004). Agent approach for service discovery and utilization. In *37th Hawaii International Conference on System Sciences*, Hawaii.

Peltz, C. (2003). Web services orchestration and choreography. In *Computer*, volume 36.

Schmeck, H. (2005). Organic computing a new vision for distributed embedded systems. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC05)*.

Syntec, I. (2007). *Les architectures orientes services*. collection thematic, France.

Zhao, L., Mohan, J., and Liliang, L. (2007). Services computing as the foundation of enterprise agility: overview of recent advances and introduction to the special issue. In *Information systems Front*. Springer Science.