Agent paradigm and services technology for distributed Information Sources

Hakima Mellah
Research Centre in Scientific and Technical Information - CERIST, Algeria
hamellah@yahoo.fr

DOI: 10.20470/jsi.v2i4.96

Abstract: The complexity of information is issued from interacting information sources (IS), and could be better exploited with respect to relevance of information. In distributed IS system, relevant information has a content that is in connection with other contents in information network, and is used for a certain purpose. The highlighting point of the proposed model is to contribute to information system agility according to a three-dimensional view involving the content, the use and the structure. This reflects the relevance of information complexity and effective methodologies through self organized principle to manage the complexity. This contribution is primarily focused on presenting some factors that lead and trigger for self organization in a Service Oriented Architecture (SOA) and how it can be possible to integrate self organization mechanism in the same.

Key words: Agility; complexity; service discovery; self organisation; information semantic; MAS interactions protocols; SOA

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 [28][39]. 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 requestors [4]. However a service can be changed and may evolve over time [23]; a new version of a service can replace 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 [18]. In order to ensure this agility and allow use of services required for business process and user, it is important to have a service oriented software architecture (SOA) [18][28]. During the interaction, the meaning carried by information is related to the information itself, with respect to the manner of its usage 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 inter-connected information sources are considered.

The complexity theory provides a conceptual working environment, a way of thinking and a way of seeing the world [8][29] or just the information. This new approach requires new connections to introspect a new function [8] for a portion of an spatially located entity. Organization is looked as a complex evolving system, co-evolving [8] 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 it should be done, without any external entity decision [29]. Emerging properties, patterns or structures appear from the interactions of individual elements.

It is better to contemplate a SOA based system by self-x properties (where x can be presented through the terms like self-organization and self- healing) that make it self-organized without any external intervention.

The rest of the paper is organized as follows: Section 2 describes background of SOA and its corresponding relevance of its structure. Section 3 elaborates distributed information sources and its associated service, content, use and architecture in the subsections. Section 4 provides the detailed approach of SOA in the context of self organization with impact and example in practical scenario.
Section 5 presents some implementation aspects and future research background is discussed in section 6 followed by the conclusion in section 7.

2. Background

The current works on agile information sources is tend to focus solely on the quality of the information requested by the user in terms of the representational content. Ontologies are proposed and exploited for this purpose [40]. Agile information systems are characterized by the appearance of evolving represented services. Only this evolution and exchange between distributed information sources happen in an environment, where we observe the following limitations:

- The approach about the usage of the information
- The consideration of the environment where the information exchange and information request takes place

The semantic [37] [31] is a way to find relevant information and integrate it in the one link. The semantic approach helps in research, discovery, selection, composition and integration of Web services. The ontology [40] is a key technology behind the Semantic Web, by adding more knowledge to information, it makes it more meaningful.

In our approach, the relevant information is not identified only by its contents, which can find a semantic in ontology but also in an environment that supports other contents that are connected through an information network and are characterized by different usage. The contents that characterize an information source are also used for one or more uses, in relation with other uses and other contents as well. Keeping this consideration in mind, we apprehend the relevant information in a complex manner to propose a relationship based representation model of distributed information sources (in a network) based on a three-dimensional model incorporating the content, the use, and the structure. At the same time, we consider the information system according to the services provided to the user (customer needs), and initiate kind of mapping between these two aspects. The idea is to provide a vision/model incorporating these different aspects. Our model is based primarily on the multi-agent systems paradigm and a SOA.

Compared to traditional distributed approaches, SOA serves business demands in a more efficient and organized way by applying a set of design principles, such as service abstraction, reusability and composition. 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 [11]. However self organization increases agility in SOA as it gives it robustness. Gunasekaran [16] considers enterprise agility as the ability to survive and prosper in a changing environment. When an organization provides services, its information system interacts not only with a set of diversified services and requests, but also with a variety of companies (other organizations) to provide/request services. During the information exchange, the following concerns are underlined:

- A service can be changed and may change over time, as new versions can occur with best characteristics in terms of cost and quality.
- A service cannot respond any more
- New organizations can integrate the interaction process by introducing new services within their information system.
- Some organizations may also withdraw the interaction process.

Web services reliability is strongly dependent on the fault handling mechanisms of the communication protocols and on the messaging infrastructure mediating their interactions [13]. Correct service delivery continuity defines web services reliability. This implies zero, or at worst, relatively few failures and rapid recovery time [13]. WS are managed by different providers geographically distributed. The unreliability of any of the constituent WS could lead to the failure or QoS degradation, even if other constituent happen to be reliable. A service management middleware is proposed in [13], it is based on some recovery policies (eg rety, skip, use). In [28] a framework is proposed for WS orchestration, as environment of this latter could present some exceptions that are not possible to identify precisely their causality. WS are used to diagnose exceptions in a more precise manner. In [2] an interaction protocol for every cooperating web service WS is represented by an abstract process. The local view of WS, on choreography is determined by associating its abstract process to the final WS. In our approach, the interaction protocol is supported by a multi agent system assuring self organization, when a dysfunction or anomaly is suspected.

Therefore, the problem statement is summarized in the following points:
• How to contribute to enterprise information system (organization) agility characterized by content that may be required for different uses, knowing that this content can be in relation to other ones and other uses?
• How to organize/reorganize these contents in connection with the uses and the services they offer, without being visible to the user?

3. A representation model for distributed information sources

3.1 System functioning

Figure 1 represents the functioning process of the proposed system. The user expresses its request for an informational required on a given entity. The latter encapsulates the contents that have a spatial localization. After an analysis and semantic interpretation, following the use of concepts ontology, and on the basis of relative uses characterizing contents, the latter can be decomposed into other contents. Contents will correspond to a basic service, only if it is executable. It is decomposed relatively to its relative uses, until it will be able to give an answer to the user, and thus accomplishing its goal. The user offered services can be simple or composite. The composition or decomposition of services is beyond the scope of the present model. There are substantial works have been reported on the subject [4] [38] [41]. Contents decomposition basically incorporates the relative use to generate a structure (or pattern). There are different services composition platforms, such as StarWSCoP [11] Self-Serv [5]. The needs of the users are identified and decomposed into abstract services through the Service Composition Engine. After sending a request to SOAP (Simple Object Access Protocol) service register, concrete services are returned and executed. The service and context concepts are defined in next subsections.

3.1.1 Service

A service is a clearly defined function, independently from technical platform, which supports it and also supports the status of other services. A collection of heterogeneous services can be used to run composite applications or business processes. A service can be simple or composite [11]. The composite service includes a set of features. For example, relatively to accommodation context "hotel solutions" content is a composite service that provides “catering” and “accommodation” as the basic services.

3.1.2 Context

The use of a service not only depends on its features but also on its use context [11]. At this point, information is attached with a set of attributes and a purpose. The purpose is the goal, for which the context is used at a given time. The finality is viewed as the information that is most appropriate for the agent (software or human) at any given point of time. In the example below, according to the accommodation context with the attributes roomtype, surface, sight, the purpose is to consider the contents on the accommodation in order to offer the user accommodation services, and the finality is to decide whether to accept a room booking or not. While context can be seen as a container, holding information relevant to the problem or domain being examined [6], a context aware application is defined as one that uses the context of an entity to modify its behaviour to best meet the context of the use [11]. Since, the context is being considered during the performance of an activity or a set of tasks by an agent, hence, the context will only be into existence, when the activity is being performed. It is created at the beginning of an activity and is used throughout the performance of the activity, and exits when the activity terminates [10] [36].

Context is discussed in [34] [16] as a concept of dimension. It describes the "external" dimension ('physical'), where the context can be measured by hardware sensors i.e. location, sound, movement, temperature, air pressure, etc. The 'internal' dimension ('logical') is essentially specified by the user or controlled through its interactions, i.e. user’s goals, tasks, work context, business processes and emotional state of the user. In addition to the "external" context, the logical context is considered for summarizing the user query or tasks (Figure 2). In general, tasks could be any process such as solving a problem, reaching a conclusion, making a decision, answering a question or taking an action. For web services in particular, things e.g. selecting a service and creating a service composition are typical tasks [21]. We consider a context attribute as a function, with one or more parameters, returning a value. For instance, context attribute "RoomCost" is a function defined on the set of Rooms, returning a float value corresponding to the price of a Room. If A is a set of all attribute a, and v is a set of values, Va is the definition domain of a, (example: Vtime= [0, 24]), Pa is a set of
parameters needed to compute the value of $a$. We call an instantiation of context attribute $a \in A$ as a pair $(a,v)$ where $v$ is the set of values $v \in P(V_a)$ of $a$ at a given moment. For instance, $(\text{Day}, \{15\})$, $(\text{Roomtype}, \{\text{Single, suite}\})$, $(\text{roomsight}, \{\text{seasight}\})$ are instantiation of respective context attributes Day, Roomtype and Roomsight.

Figure 1. Information's handling and execution process relatively to content, use and structure.

3.1.3 User request
A user request is a coordinated set of activities that can realize the needs requested. For example, a query regarding a Request for accommodation following a business trip may include activities such as the hotel accommodation, car rental, planning meetings, and so on.

Figure 2. Modelling relationship between service, content and context considering user's tasks

The request is described in terms of services ontologies and can be represented using state-charts [7]. A state-chart, representing a user task, consists of states and transitions. A state can be simple or composite. A simple state can be annotated by an operation of a concrete service or an operation
defined in the category of service ontology. In the latter case the mapping of the activity to the service will be executed in run time.

While the user introduces his query, a service discovery operation is set off. Therefore, by selecting and introducing some attributes such as service attributes and context attributes, the user refines the discovery. During the execution of discovery operation, a pattern is generated. It is constituted by all the services that can answer the user request and which are assimilated as the information across network nodes. At a task definition time, the administrator associates each query with its corresponding logical contexts, where as physical contexts are captured by sensors (e.g., for a travel request, time zone, local currency may be relevant contexts). Interactions relevant context (logical context) attributes are captured through the user and may be for example for hotel accommodation: Roomcost, Roomtyp and Roomsight.

### 3.2 Concepts and motivation

In order to consider the user of the system and through the services expected, it seems necessary to apprehend the information in a complex manner in a three-dimensional system. Our approach therefore focuses on the identification of three parameters governing the complexity of the information that are the content, the use and the structure, in the enterprise information system (organization). The enterprise information system draws its resources from different and distributed information sources. Our contribution to the agility of an enterprise IS aims at:

- Satisfying the user's evolving needs for the services he expects from it, and ensuring a better quality of service (QoS).
- Having the ability to organize/reorganize the contents of information related to the uses, without involving the user in this task.
- Being able to make the best use of related information, highlighting an organisational structure.
- In the next sub-section, we elaborate all the definitions of necessary concepts associated in the modelling of our approach.

#### 3.2.1 Content

A content is an information source or a set of elements participating in information management, processing and diffusion within an organization. We mention about basic content (simple) or composite content. When the content is basic (corresponding to a “business function”), it disseminates information in the form of services (computer services), however when it is composite, it is expected to be decomposed into other content relatively to related uses until reaching the last basic content. The goal is achieved through one or more contents. For instance, a hotel solutions company presents its services through a "hotel solutions" entity which represents content “C”.

#### 3.2.2 Use

We distinguish two types of use:

(i) **Global Use (GU) of Information source (IS)**

The user informational needs demand an organizational strategy, as well as the use of different contents from distributed information sources. The way that content is used, and the goal achieved through this contents discovery are expressed as a global use. For example, a global use of C content may be information retrieval or simply surfing.

(ii) **Relative use (RU)**

RU represents a feature of content use. However, it may consolidate a set of functionalities having a common goal that is identified as the use context of content. For example, "Hotel solutions" entity can be used to its "catering" context instead of "accommodation" one, so "catering" and "accommodation" as two relative uses of C content.

There is a strong correlation exist between the context notion and the relative to characterize a content, and between the content of an IS and a service. In order to answer the user's query, the information was framed in three important concepts that are: service, context and services discovery structure. Figure 3 shows all the parameters involved to enrich the semantic of information in a UML diagram.

#### 3.2.3 Structure

Relative uses of characterizing the contents offer the possibility of its decomposition. However, a necessary content for user's query may be available at another IS, located on the information
network. Responding to the user’s needs generates a decomposition structure that is identified as an organizational structure/pattern. The structure’s nodes correspond to different contents involved upon decomposition. The bonds are considered as the relative uses.

In order to take into account the three parameters (content, use and structure), for modelling information of an agile IS, we have been constrained to use the services technology to take benefit from their composition and discovery mechanisms. The example below illustrates the concepts given above.

### 3.2.4 Example

Let consider the “hotel solutions” enterprise; at a high abstraction level one can see this entity with different contexts, in the sense that it can provide the user with solutions that do not appear at first glance. Three solutions are considered to be appropriate to different contexts, which are hotel accommodation, staff management, resources management and catering management.

![Diagram](image)

**Figure3. Interpretation of information decomposition problem by content, use and structure with UML.**

Depending on the context chosen by the user and the attributes characterizing it, the decomposition of “hotel solutions” entity content will generate a structure recognized as the appropriate organization of the considered entity for a moment.

When the user introduces some attributes in a context characterizing a service, a context summary query is specified based on a conjuncture query (using the operator “\&”) involving the context attributes. Some parameters useful for the service selection process are required such as service attributes, service operations inputs/outputs.

1. **Accommodation solutions**

In accommodation context, the entity hosting solutions includes, all the necessary features to book hotel rooms, car rentals, as well as some functionality encapsulated in the “diverse” entity representative of the services offered by the hotel. To use the “hotel solutions” entity contents, some scenarios are illustrated, such as hotel booking, job hunting, we consider as an example the booking hotel room and recruitment scenarios:

- **Scenario1 “Room hotel accommodation”**.

The scenario illustrates a user who comes to book in a hotel. In this case the use context of the entity is solutions hotel accommodation. It is also considered as the internal dimension of this entity, and its attributes are for example roomcost, roomtype, roomsight .. At each of these attributes corresponds a value introduced by the user. The use context of “hotel solutions” entity could be the recruitment and its attributes could be for instance: jobrequested”, “attributedsalary”, “requestedsalary”...

- **Global use(Gu): “user informational needs”**

To answer the user’s query, a communications structure (pattern) is generated (figure 4), the structure is composed by a set of nodes and links, corresponding to the content and related uses.
They are considered as a mean of decomposition in depth and as a mean of interaction in width. All the nodes involved in this architecture are the:

- user
- hotel solutions entity
- composite Content "accommodation solutions" (C)
- "single room booking" content (C1)
- "hotel car rental" Content (C2)
- "Diverse" Content (C3);

the links involved are:

- Ur1: relative use of C content; connects C to C1.
- Ur2: relative use of C content, connects C to C2

3.3 System Architecture

The proposed system architecture is based on the Multi Agents paradigm and on the new services technology, and focuses mainly on the structure that is being produced to satisfy the user in his request. The latter can be done locally, i.e. through existing services within the prompted organization, or globally i.e. a search for some activities. They can be launched in the information network linking the prompted organization with those who share their activities for it. For this purpose, mechanisms for selecting and matching services will be exploited. The agents that are required to carry out this work are: the user agent, the context agent, the notify agent and the service agent.

(i) The user agent or UA for short interacts with two types of users: An administrator in order to define, modify and save tasks and an end user to view, select and execute the tasks. Once the user selects a task, the user agent:

- Contacts the agent context for searching context
- Receives context information
- Asks the service agent to offer the most appropriate services to the chosen service. The service agent transfers its recommendations to the user agent after consulting service ontology.
- Decides which services to make running,
- Interacts with the service orchestration engine to perform the task by invoking the necessary services and orchestrating all their interactions.

Services selection and aggregation mechanisms are being exploited, they can extend even beyond the considered organization. The generated pattern constitutes the structure, by which it has to be channelled service discovery to respond to the user request. The structure includes nodes for characterizing services and links that connect nodes in different context.
The context agent or CA is powered by a provider of contexts. It assists the user agent and helps to facilitate the location of the service requested by the user. The provider of contexts assures its generation, its attributes and their values.

The Service agent, namely SA is powered by the service provider and requested by the context agent to submit relevant services to a desired context. The SA presents the services that correspond to the submitted contexts after consulting the service ontology.

While the service provider submits information to the service agent, the user agent may request a service that has already been used previously. The role of the Notify Agent (NA) is to inform the UA on the existence of the requested service, but with advanced or better features. The user can, however, change its demand on the service.

For implementing the system, a layered architecture is proposed. Figure 5 represents all elements of the architecture that are grouped into layers: i) A user layer, ii) Information source layer, iii) Infrastructure layer.

User layer: It is the most important layer of the system. The user interacts with the user agent through this layer. The administrator uses this layer to define, store and maintain tasks. The user task is the request he introduced to get his needs. The layer provides task editor to describe the task state chart. With the tasks editor, the user can annotate a task with contexts and services ontologies.

Information source layer: It is the intermediate layer between user and infrastructure layers. The components of this layer are information sources or content that the user wants to prompt through a graphical interface in the form of portal. The content is encapsulated in Web service. A localized content on a portal may not respond to a request user. The user agent will be charged in this case to ask the service discovery of the infrastructure layer, to find other content recognized as services that can satisfy the user request. To do this, the service discovery will use the context information.

Infrastructure layer: This layer is composed of a set of tools for the discovery, location and execution services, it interacts with the upper layers as well as service and contexts providers.

4. 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 [20]. Using service orientation in the architectural design facilitates reusability, flexibility, interoperability, and agility of this kind of systems.

Figure 5. A system layered architecture combining agents and services technology.
Generally, service autonomy raises the question of how to establish proper operation status on the system level [20], 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 behavior 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 [18]. This definition applies also to the many facets of self-organization called self-x properties [35]. For example, a system is self-healing, if it can eliminate the effects of mal-functioning 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 [35]. 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 [27].

4.1 Complexity theory and SOA

Organization theory deals with complexity as a structural variable that characterizes organizations and their environments [22]. Daft [9] matches complexity to the number of activities or subsystems, which basically composes the organization. It can be decomposed into three dimensions: vertical, horizontal and spatial, corresponding respectively to the number of levels in the organizational hierarchy, the number of posts or departments within an organization, and the number of geographical locations. We can identify the following understandings:

- Activities/subsystems are only the services that a SOA can offer to its users;
- activities/services have a geographical location that may well change, so they do not have a persistent physical location which is a characteristic of service-oriented approaches [15];
- activities/services can be decomposed to highlight features not visible at first sight. While they are decomposed, a service discovery structure is generated.

The points highlight the key features of agile information systems, well supported by a SOA. The key features appear to be compatible with Daft's proposal [9]. In order to tackle information system complexity, it is required to harmonize the quality of service offered to users internally and externally. A new version for the design of complex systems is called as organic computing.

The latter satisfies conventional requirements for trustworthy systems, which adopt autonomously to dynamic changes of the environment, and have self-x (self organization, self healing) [35] properties as postulated for Autonomic Computing.

From our perspective, we recognize the information complexity, issued from a set of interacting information sources or just to investigate a SOA, in a three-dimensional system handling service, context and service discovery structure. Each IS represents one or more organizations. We can note the presence of a mapping between our work and that of Daft, interpreted in the following points:

- At each level of an organizational hierarchy correspond to the existence of services, and the number of departments in an organizational level may correspond to the number of contexts that we ascribe to service.
- The geographical locations constitute the service discovery structures, generated by the various services discovery, to meet the needs of the user and achieve the desired goal.

Considering these technicalities, we confirm that complexity theory is easily supported by a SOA. Consequently, there must be an issue to integrate self organisation within a SOA as it is a key feature of complex systems.

4.2 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 [33].

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 [33], choreography describes the collaboration between a collections of services, whose aim is achieving a given objective. The
achievement of this objective is done by exchanging messages [3]. Therefore, in choreography it is possible to have multiple orchestrations and in each one, one service will act as an orchestra chef. In [26] 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 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.

4.3 Why is it important to integrate self organization in a SOA?

In services discovery process, information retrieval includes two orthogonal components [26] which must be complementary and that are:

(i) The construction of the network: is a self-organizing process which requires an adaptive behavior while respecting the network consumption.
(ii) Research: is ensured by a distributed algorithm which will exploit the structure of the emergent network.

In [26] [18] 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 [13]:

(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.

4.4 How self organisation 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 [32] 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 [2] 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 [1]. 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 inter-related 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. Chatle [14] 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 [30] 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 [14] 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 [25] and a self organizing protocol [27] 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). As the first layer, in the proposed SOA, that interacts with users, is based on agents, we suggest [24] the recourse to a Multi Agent Systems (MAS) self organizing protocol for service discovery. The self organizing protocol is inspired by bacteria colony life.

5. Implementation

Tasks are defined by state_transition diagrams as in [7] (figure 6). The case study that is considered in this work is a tourism enterprise also known as a tourism agency that the principal task is labeled “Hotel Solutions”, it is composed of three tasks: « FlightBooking », « HotelBooking », « CarRental », each task is composed of activities and transitions between activities. An activity references an operation defined in a service category which is defined in a service ontology. Web services that are invoked depend on user preferences (Figure 7), that represent the use context, and that attributes are: Origin, destination, Class Seat(A/B), Price,...etc like shown in (figure 7). However two different users (different preferences) executing the same task can receive different results. Three types of web services are affected to « Hotel Solutions » task, that are: « FlightBookingService », « HotelBookingService », « CarRentalService ».
Figure 6. An example of FlightBooking task model

The user query is analyzed by (Java Server Pages) JSP (Model (Figure 8) that architecture is also known as MVC (Model, View, Controller) design. By the mean of this model, the controller (implemented by a servlet) analyzes the http query. Based on application logic, the controller generates the necessary Java Beans. Once the task identified by "IDtask" is executed, the controller asks the SMAWrapper the necessary task's attributes list (SMAWrapper.retrieveTaskattributes(IDtask)). SMAWrapper asks the user the corresponding attributes by the mean of (Agents Communication Language) ACL message.

The query is the summarized as a set of pairs (c,v) where c is an context attribute and v is the value (e.g., (origin,"houari boumedienne"),(destination," ") (seat class, "Class A")). The Context Agent delivers the needed attributes. For service discovery SMAWrapperreeseachservice (preferences) is called. The Gateway Agent translates the query into ACL message that allows the Service Agent to launch the search into the (Universal Description Discovery and Integration) UDDI.

Once the services are found, their URL(s) are saved. Service Agent sends the services list to UserAgent and then to the controller by SMAWrapper. The controller redirects the query to listservices.JSP that generates the services results. After a selection of services (Figure 9) to be composed, the controller redirects the selection to the composition Engine (Business Process Execution Language (BPEL) related to the task (BPEL file). The composite service is executed, and the results are transferred to Results.JSP.
6. **Future research directions**

While discovering services, the generated structure must be self organized in order to achieve a sort of robustness within the global system. A lot of research works deal with services discovery and some of them are MAS based. In Palathingal’s[30] 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. It is foreseen to propose a MAS for services discovery inspired from [30][13], and principally based on data duplication, within the service’s provider. Therefore, instead to affect a provider of services to a single machine, services can be duplicated on several machines and at each one a software agent is associated with it.

---

**Figure 7. User preferences form for holiday booking’s task that generates an html code, which will be browsed to the user (Figure 10).**

**Figure 8. System Software Architecture**
At each provider will correspond any more or there is a resource bottleneck, another several machines. When a service does not respond 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 [27], and failures are detected by the checking primitive that keeps alive interactions between agents machines [24]. 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.

Figure 9. Selection form for Web services execution

Figure 10. Execution results form
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.

7. Conclusion
The research of contents that is expressed by the mean of tasks introduced in the system is interpreted as a service discovery process, for achieving profit from their composition and discovery methods. We consider in a first time that a content, that represents an information source, is encapsulated into a web service. So, there is a structure (a pattern) that is generated through decomposition of contents characterized by their relative uses.

To treat the information semantic problems, a concept ontology relating to web services and contexts can also be referenced and used. The semantic of information is not treated simply by ontology; it is considered and modeled in a three dimensional system where dimensions are content, use and structure. It is shown that while a distributed content is decomposed, a pattern is generated and identified. Our contribution to the agility of the enterprise information system offers to the generated structure robustness and means to adapt to unforeseen changes. The structure robustness and agility are also interpreted by making the system more adaptive towards the dynamic changes of its environment (discussed in section 4) such as services failures. This is done by the attribution of self organized properties to the system and particularly to the generated structure. The proposed architecture puts into evidence that agent’s layer deals exactly with the integration of a self-organizing protocol in the context of service failure. Finally, the proposal presents the evidence of self organization of the content discovery which is based on service discovery structure and ensures a total discretion towards the user.

8. References


[34] PREKOP, P., & BURNETT, M., Activities, context, and ubiquitous computing, Special issue on ubiquitous computing computers communications, 26 (11), 2003.


**JEL Classification:** L86