Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 14(2) June 2013

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 14, Number 2, June 2013

TABLE OF CONTENTS

Special Issue on Multi-agent Systems: Methodologies, Tools and Applications:

Introduction to the Special Issue **iii**
Viorel Negru and Daniela Zaharie

A Swarm-inspired Technique for Self-Organizing and Consolidating Data Centre Servers **69**
Ionut Anghel, Cristina Bianca Pop, Tudor Cioara, Ioan Salomie and Iulia Vartic

Multiple equivalent simultaneous offers strategy in an agent-based grid resource brokering system – initial considerations **83**
Naoual Attaoui, Maria Ganzha, Marcin Paprzycki, Katarzyna Wasielewska and Mohammad Essaaidi

An Agent-Based Approach For Hybrid Multi-Cloud Applications **95**
Djamel Benmerzoug

Establishing semantic closeness in an agent-based travel support system **111**
Mariusz Mesjasz, Marcin Paprzycki and Maria Ganzha

Designing a Scalable Social e-Commerce Application **131**
Eugenio Zimeo, Gianfranco Oliva, Fabio Baldi and Alfonso Caracciolo

DOI 10.12694/scpe.v14i2.840
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON MULTI-AGENT SYSTEMS: METHODOLOGIES, TOOLS AND APPLICATIONS

Dear SCPE readers,

Autonomous agents are software entities that are capable of independent action in open, unpredictable environments. Agent technologies include models, frameworks, platforms, methodologies and tools for developing agent systems. Agents and multi-agent systems are currently applied in diverse domains that require complex problem solving, e.g. e-business, e-learning, e-government, e-health, social networks, risk management, planning, distributed computing, simulation, optimization, games etc.

This special issue collects papers on applications of agent-based systems in various fields (resource management, interaction protocols, decision support systems). Four of the papers included in this issue are extended versions of papers presented at some conferences and workshops related to sofware agents: WASA 2012 - 2nd Workshop on Applications of Software Agents, WIMS 2012 - International Conference on Web Intelligence, Mining and Semantics, IDC 2012 - 6th International Symposium on Intelligent Distributed Computing. Even if the papers included in this issue address different problems and use different technologies, all of them rely on concepts, paradigms and techniques specific to agent-based modelling.

The first paper proposes a technique, based on swarm intelligence, for self-organization of data centres servers in order to reduce the power demand. In the proposed approach, each server is modelled by an intelligent agent, member of a decentralized swarm of cooperating entities. The simulation results show an increase of the hardware utilization ratio and a decrease of the power consumption.

The second paper is related with the *Agent in Grid* project aiming to integrate Grid and agent-based systems. The paper proposes a strategy for multi-attribute negotiations within an agent-based Grid resource management system.

The third paper presents a formal framework based on agents for designing and verification of interaction protocols. The agent-based system has been designed to enable interoperability and cross-Cloud application management.

The fourth paper reviews the current research an agent-based travel support systems and proposes an algorithm for semantic matchmaking between the user profile and the instances of an ontology.

The fifth paper proposes an architectural pattern in designing scalable intelligent enterprise systems. The proposed pattern has been experimented in eCommerce applications by designing an intelligent and scalable virtual mall.

Viorel Negru and Daniela Zaharie
West University of Timisoara, Romania

DOI 10.12694/scpe.v14i2.841
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. 69–82. http://www.scpe.org

A SWARM-INSPIRED TECHNIQUE FOR SELF-ORGANIZING AND CONSOLIDATING DATA CENTRE SERVERS

IONUT ANGHEL, CRISTINA BIANCA POP, TUDOR CIOARA, IOAN SALOMIE, IULIA VARTIC *

Abstract. This paper proposes a swarm-inspired data centre self-organizing and consolidation technique which aims at reducing the power demand in data centres while ensuring the workload execution within the established performance parameters. Each data centre server is managed by an intelligent agent that implements a bird's migration-inspired behaviour to decide on the appropriate server consolidation actions. The selected actions are executed to achieve an optimal utilization of server computing resources thus lowering power demand. The data centre servers self-organize in logical clusters according to the birds V-formation self-organizing migration model. The results are promising showing that by using the proposed swarm-inspired solution, the data centre Deployed Hardware Utilization Ratio Green Indicator increases compared to the widely used Fit First consolidation algorithm. The average power saving of the proposed technique is around 40% of the power demanded by the data centre computing resources and about 16% of its total power demand including the IT facility, when comparing to OpenNebula Fit First consolidation technique. This paper is an extended version of the one published in WIMS'12 proceedings showing more details about the swarm-inspired consolidation technique and the defined algorithms.

Key words: Swarm optimization, self-organizing, data centre, server consolidation, birds V-formation.

AMS subject classifications. 68T42, 68T05, 37N25

1. Introduction. Green computing is receiving more and more interest from both data centre owners and research groups. Green IT is defined as designing, manufacturing, and using computer systems in an energy efficient manner with low or no environment impact [1]. Initiatives such as Climate Savers Computing Initiative [28] or the Green Grid [38] aim at developing methods and techniques for reducing the energy consumption in computing environments. Studies have shown that exponential growth trends of the data centres are no longer sustainable without the careful consideration of their energy efficiency [29] [31]. An important rising concern is the environmental impact in terms of carbon dioxide (CO_2) emissions caused by high energy consumption. From this point of view, CO_2 emissions of the worldwide data centres are equivalent to about half of the total airlines' CO_2 emissions and are expected to grow from 76 $MTCO_2e$ to 259 $MTCO_2e$ by 2020 [32]. Studies show that the cost of the energy required by a data centre server along its lifetime will probably exceed its hardware costs [18]. Data centre electricity consumption accounts for almost the 2% of the world production and their overall carbon emissions are greater than both Argentina and Netherlands [25]. Moreover, infrastructure systems such as cooling systems or power delivery systems supporting the IT equipment itself, consume 60% of a data centre's energy usage. All these problems led to considering energy or power efficiency as first-order objective in the design of modern computing infrastructures including data centres and clouds.

In this context, many researches from both management and technical background, are working to reduce the energy consumption of IT data centres. In general, data centre administrators are always focusing on the performance aspect when configuring the data centre servers or clusters and ignore the energy consumption. The sub-optimal utilization of computing resources is one of the main factors reported in the literature that contributes to the data centre high energy consumption [25]. To solve this problem, state of the art solutions use techniques based on virtualization and server consolidation for minimizing the number of underutilized servers in data centres [2].

The problem of organizing and using the data centre IT hardware resources in an energy/power efficient manner can be approached as an optimization problem. Nature can serve as an inspiration for solving different domain problems. Biological systems feature organized, complex and intelligent behaviour. Applying biology concepts in the field of computer science is a new evolving research direction. The main idea is to model and develop computer systems by mapping biological principles and concepts to the actual problem that needs to be solved by computer science. By inspiring from insects, birds or animals behaviour, efficient and self-managing distributed systems can be designed and modelled. Thus, the problem of organizing and using the data centre

*Computer Science Department, Technical University of Cluj-Napoca, Romania ({Ionut.Anghel, Cristina.Pop, Tudor.Cioara, Ioan.Salomie}@cs.utcluj.ro).

IT hardware resources in an energy efficient manner can be approached as an optimization problem and can be solved by applying bio-inspired concepts and heuristics. Birds and insects are both employing swarm level self-organizing behavioural strategies, which help them to live and survive by efficiently using and conserving energy. For example, penguins self-organize in swarms in case of cold temperatures and high winds by huddling themselves in concentric circular formations [22]. In these formations, the penguins which are not on the exterior side are kept warm, leading to energy conservation. The exterior penguins are the ones that face the harshest conditions and ensure the comfort of the other members of the colony, but they are continuously replaced by other penguins coming from the interior part of the formation. The swarm of birds' flying in V-formations during their migration is another self-organizing behavioural strategy for conserving energy [23]. In such formations, the leading birds (situated in the head of the V) together with the ones situated in the tips of the V-formation wings, consume a lot of energy while the other birds consume less energy [24]. In this context, by inspiring from the swarms' self-organizing behavioural strategies which have ensured the survivability of different biological species over time, energy efficient self-organizing strategies for data centres servers can be developed.

This paper proposes a server consolidation technique for reducing power consumption in data centres inspired from the birds self-organizing behaviour during migration. Each data centre server has an attached intelligent agent which implements behaviour similar with the behaviour of a bird in the V-formation. The V-formation leading birds are modelled by a cluster of active data centre servers that can accommodate the incoming workload. The birds from the start of the V-formation wings are represented by a cluster of data centre fully loaded active servers that are candidates for powering down after finishing the execution of their deployed workload. The birds in the middle of the V-formation wing are modelled by a cluster of servers containing powered down servers, while the following birds in the V-formation wing are represented by a cluster of idle servers that are candidates for passing to active states and for accommodating the incoming workload. The servers attached intelligent agents communicate and collaborate to decide on the following types of actions: resource consolidation actions (workload deployment and migration), dynamic power management actions (turn on/off server) and server placement in the appropriate V-formation cluster. This paper is an extended version of the paper [30] and shows more details about the swarm-inspired consolidation technique and the defined algorithms.

The rest of the paper is organized as follows: Section 2 presents related work, Section 3 introduces the swarm-inspired consolidation technique, Section 4 describes a case study and relevant evaluation results while Section 5 concludes the paper and shows future work.

2. Related Work. Organizing and using the data centre IT hardware resources in an energy/power efficient manner by applying virtualization and server consolidation techniques is a NP computational optimization problem which can only be solved by using a holistic approach.

The current approaches to resource consolidation take advantage of virtualization by proposing models to migrate the virtual machines in data centres from one server/cluster to another [20]. By virtual machine migration, the workload can be consolidated on a smaller number of physical machines allowing for servers, or even for entire operation nodes, to be completely shut down [27]. Authors reveal that when consolidation is used, an optimal solution for energy/performance trade-off can be defined. Efficient consolidation models based on the bin packing technique were proposed in [10]. Two well-known heuristics for the bin packing based consolidation, the best-fit decreasing (BFD) and the first-fit decreasing (FFD), were used [19]. To enable energy efficient consolidation, the inter-relationships between energy consumption, resource utilization, and performance of consolidated workloads must be considered [9].

In [15] self-configuring and self-optimizing agents handle user requests to dynamically allocate resources for cloud applications. Support Vector Machines and Model-Predictive Control are employed to predict and plan future virtual machines configurations while guaranteeing SLAs. The agents share a common knowledge base where adaptation rules are stored. A thermal aware workload scheduling and consolidation solution aiming to reduce the power consumption and temperatures in data centres was proposed in [12]. The simulation results show that the algorithm can significantly reduce the energy consumption with some degree of performance loss. In [25] a novel technique for controlling the data centres servers CPU allocation and consolidation based on first order Kalman filter is presented. In [14] the server consolidation problem is approached for small data centres as a constraint satisfaction problem. The authors also propose a heuristic for dealing with server consolidation

in large data centres.

Learning techniques are also used to trade-off between computing resources power consumption and performance during the allocation process [16]. In [17] the problem of dynamic server consolidation in virtualized data centres is approached by proposing the development of an energy aware run-time consolidation algorithm based on reinforcement learning. In [11] a consolidation methodology that uses machine learning to deal with uncertain information is proposed. Previous server behaviour data is used to predict and estimate the current power consumption and also to improve the scheduling and consolidation decisions.

Few state of the art approaches study and use biologically-inspired techniques for optimizing the energy/power consumption in IT systems and data centres. Since biological systems naturally tend to conserve their energy, many simple principles found in the biological systems might be used in IT power management [4]. The adoption of biological principles (e.g. decentralization, natural selection, symbiosis) in the process of designing and building services on top of server farms is proposed in [6]. A service is designed as a biological entity, equivalent to an individual bee in a bee colony that competes or collaborates for computing resources. Using natural selection principles, the services that waste energy are banned for execution. An immune-inspired method for designing applications capable of adapting to network dynamic changes is proposed in [8]. The environmental conditions are modelled as antigens, while the agents (used to design applications) are modelled as biological entities having an immune system.

In [5] the management of energy consumption in wireless sensors networks is approached using a biologically-inspired agent based solution. The agent behaviour focuses on biologically-inspired actions (e.g. pheromone emission or migration), each of them having an associated energetic cost. Through these actions, the life time and the state of each agent evolve autonomously and there is no need of a centralized control unit. Inspired by the behaviour of insects in search for a proper migration place, the authors of [7] propose a method for optimizing the energy consumption in data centres. The migrating insect is modelled by a running virtual machine, a colony of insects is modelled as a set of virtual machines running on the same server, and the candidate migration places are modelled by servers. Authors use a scout-worker migration model in which a set of agents (scouts) investigate each server to identify the appropriate one where a virtual machine can migrate. In [3], authors formulate the problem of server consolidation as an optimization problem which aims to maximize the number of servers hosting zero virtual machines, while ensuring that all available virtual machines are run. To solve this optimization problem in a decentralized and self-organizing way, authors propose a gossip-based algorithm in which each server interacts with its neighbours by means of messages containing the number of virtual machines running on it. The server that has space for extra virtual machines accepts to host them.

3. The Swarm-inspired Consolidation Technique. This section introduces the server consolidation technique inspired from bird's behaviour during migration. The goal of the technique is to reduce the data centre power consumption while ensuring the workload execution within the pre-established parameters.

3.1. Biology Inspiration. During migration, the birds organize in a V-formation which consists of a leading bird followed by two lines of birds (the V-formation wings), flying closely behind each other (see Figure 3.1).

In this formation, the birds constantly shift positions and the leading birds periodically leave their position and re-join the line further back. Such a V-shape organization reduces the drag force that each bird experiences compared to flying alone. Also, this orientation allows the birds to communicate more easily to keep the flock together, thus minimizing the possibility of losing birds during migration over long distances. The theory behind swarm of birds V-formation energy conservation was proved by researchers through monitoring flying birds' heartbeats. Also, it has been proven that a swarm of birds in V-formation can fly up to 70% further distance than one bird, with the same effort [21].

The leading bird works the hardest due to its flying into undisturbed air. The flying of this bird optimizes the aerodynamics of the two birds following it. The two birds behind the leading bird help also reducing the drag that is experienced by the leading bird. The V-formation eases the flight of all birds through energy conservation with the cost of the bird in the lead position which works the hardest. When this bird is tired, it will leave the lead position and move into one of the lines of the V-formation from its back. As a consequence, a bird from the back of the V-formation will take the leading position, thus keeping the formation together. This self-organizing behaviour allows all birds to be leaders as well as to be in the back of the formation to save

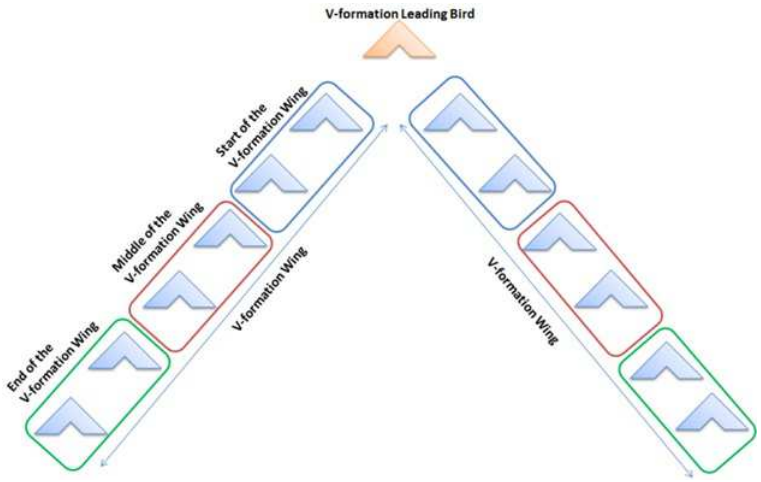

FIG. 3.1. *Birds V-formation organization during migration*

energy. The V-formation model is used as inspiration for the current approach to logically organize servers in data centres.

3.2. Concepts Definition. For developing the swarm-inspired consolidation technique, the data centre consolidation problem must be formalized.

The *data centre workload* is formally represented as a set of virtualized tasks (VTs) annotated with Resource Allocation (RA) requirements. Virtualization provides a uniform and dependency-free management of server workload tasks, while enabling facilities like virtualized task migration.

The *virtual task* is formally defined using its RA requirements for the server's processor (CPU), memory (MEM) and hard disk (HDD) computational resources as follows:

$$VT = [CPU_{req}, MEM_{req}, HDD_{req}] \tag{3.1}$$

The data centre is defined and modelled as a bi-dimensional array, C_{ij}, storing the data centre available servers and the allocation of computational resources. If the virtualized task VT_i is placed on the server S_j, then $C_{ij} = 1$, otherwise $C_{ij} = 0$.

In this context, the server consolidation problem is reduced to finding the optimal C_{ij} configuration in which the computational resources of the data centre' servers are efficiently used. Formula 3.2 shows an example of a bi-dimensional array C_{43} modelling a small data centre with three servers (S_1, S_2, S_3) that accommodate four virtual tasks (VT_1, VT_2, VT_3, VT_4). According to the C_{ij} array definition, the tasks are deployed as follows: VT_1 and VT_4 are placed on server S_1, VT_2 is placed on server S_2 and VT_3 is placed on server S_3.

$$C_{43} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.2}$$

A *data centre server* is defined by its state and the load level of its resources:

$$S_i = \{[CPU_S, MEM_S, HDD_S], state_S\} \tag{3.3}$$

where CPU_S, MEM_S and HDD_S are the current load of server's resources, and *state* is defined as follows:

$$state_S = \{ACTIVE, IDLE, OFF\} \tag{3.4}$$

In the ACTIVE state the server is turned on and runs virtualized tasks. In IDLE state the server is turned on without running any virtualized task while in OFF state the server is completely shut down.

For a server, two energy efficient, optimal resource allocation levels are defined: *low optimal resource allocation level* and *high optimal resource allocation level*.

The *low optimal allocation level*, $S_{optimal-low}$, is defined as the server resources load value for which the server does not execute workload:

$$S_{optimal-low} = [0\%, 0\%, 0\%] \tag{3.5}$$

$S_{optimal-low}$ is used to determine when the server does not execute any workload and can be turned off.

The *high optimal allocation level*, $S_{optimal-high}$, is defined as the server resources load value for which the server is most efficiently used.

$$S_{optimal-high} = [CPU_{opt_{load}}, MEM_{opt_{load}}, HDD_{opt_{load}}] \tag{3.6}$$

$S_{optimal-high}$ can be determined by measurements or by using the vendors' specification and is usually around 80% of the maximum server resource utilization capacity.

The *server computational resource under-utilization threshold*, T_{under}, is defined as a percentage of the server optimal configuration computational resources values, $S_{optimal-high}$ (for example [5%, 5%, 5%]).

For the proposed technique, the IT facility aspects of the data centre are not considered, the focus of the self-organizing and consolidation algorithm being on the IT computing resources.

3.3. V-formation Self-Organization Model of a Data Centre. Starting from the V-formation self-organization model of migrating birds for saving energy, a data centre servers' logical self-organization model to accommodate and execute the incoming workload in an energy efficient manner can be defined. For defining the data centre swarm-inspired self-organizing model, data centre related concepts and resources are mapped onto the birds V-formation concepts as shown in Table 3.1.

TABLE 3.1
V-formation - data centre concepts mapping

Birds V-formation organization model	V-formation self-organization model of Data Centres
Leading bird	Cluster of active servers
Birds immediately following leading bird	Cluster of fully loaded active servers
Birds in the middle of the V-formation	Cluster of turned off servers
Birds at the end of the V-formation	Cluster of idle servers

The V-formation organization allows for mapping the specific clusters of servers on each V-formation area and to self-organize server clusters inspired from the birds' behaviour and movements in the V-formation.

Figure 3.2 illustrates the swarm-inspired logical self-organization model of the data centre servers. In the model, each data centre server has an attached intelligent agent which implements behaviour similar with the behaviour of a bird in the V-formation defined by the swarm-inspired consolidation algorithm from Section 3.4.

The arrows in the figure represent the servers' possible movement between clusters. Using the V-formation self-organization model, the data centre servers are logically grouped as follows:

- V-formation leading bird is modelled as a cluster of *servers that are in active state*. The servers execute workload consisting of virtualized tasks but their high optimal allocation level, $S_{optimal-high}$ is not yet reached, thus new incoming workload may be accommodated.
- Birds that are immediately following the leading bird in the V-formation wings are represented by a *cluster of fully loaded active servers*. These servers have reached the $S_{optimal-high}$ high optimal allocation level and other virtual tasks could not be deployed on them. After finishing the execution of the already deployed workload, these servers are candidates for powering down.
- Birds in the middle of the V-formation are represented by a *cluster of powered down servers* which have reached the low optimal allocation level $S_{optimal-low}$ after executing their workload.

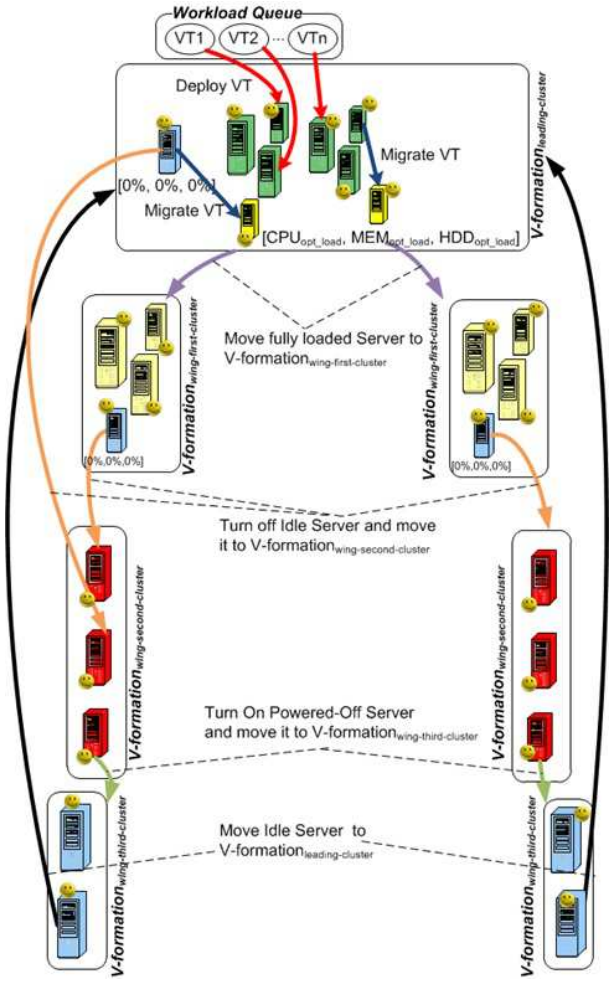

FIG. 3.2. *Data centre self-organization model inspired from birds V-formation organization during migration*

- Birds at the end of the V-formation are represented by a *cluster of idle servers* reaching low optimal allocation level, $S_{optimal-low}$. These servers are candidates for transitioning to active states and for accommodating new incoming workload when the servers from the leading clusters cannot satisfy the requirements of the incoming workload.

We formally define the *V-formation leading cluster* as:

$$V_{leading-cluster} = \{S_i | S_i.state = TURNED\ ON \wedge \exists\ j\ such\ that\ C_{ij} = 1\} \tag{3.7}$$

A *V-formation wing* is modelled by a set of three data centre clusters. The *first wing cluster* contains fully loaded active servers that are candidates for powering down after finishing the execution of their deployed workload:

$$\begin{aligned} V_{wing-first-cluster} \ = \ & \{S_i | S_i.state = TURNED\ ON \wedge \exists\ j\ such\ that \\ & C_{ij} = 1 \wedge S_i \cong S_{optimal-high}\} \end{aligned} \tag{3.8}$$

The *second wing cluster* contains turned off servers organized as a queue and is defined as follows:

$$V_{wing-second-cluster} = \{S_i | S_i.state = TURNED\ OFF \wedge C_{ij} = 0\ \forall\ j\} \tag{3.9}$$

The *third wing cluster* contains idle servers that are candidates for passing to active states. The reason behind constructing and using the V-formation wing third cluster is to make servers available on short notice to the V-formation leading cluster for accommodating new workload. This way the server wake up time delays and performance penalties are avoided. The third wing cluster is formally defined as:

$$V_{wing-third-cluster} = \{S_i | S_i.state = TURNED\ ON \wedge C_{ij} = 0\ \forall\ j\} \tag{3.10}$$

3.4. The Self-Organizing and Consolidation Algorithm. The logical movement of servers between the clusters defined by the data centre V-formation model of organization follows the birds' movements in the swarm during migration. Server's logical movement decision is taken by each server attached agent that implements the Self-Organizing and Consolidation Algorithm (see Algorithm 3.4). Each server attached agent features ticker behaviour. The agents monitor their attached server, communicate and collaborate among them to decide on the following types of actions: server placement in the appropriate V-formation cluster, resource consolidation actions (workload deployment and migration) and dynamic power management actions (turn on/off server). The algorithm takes as inputs the current monitored server state (its workload levels), a queue containing the incoming workload virtualized tasks, the values of the two power efficient server optimal resource allocation levels, $S_{optimal-low}$ and $S_{optimal-high}$, the server computational resource under-utilization threshold and the V-formation clusters, T_{under}. The algorithm is implemented as the server's agent behaviour.

[!h] Self-Organizing and Consolidation Algorithm **Input:** S - the server managed by the agent; $VT_{queue} = \{VT_1, .., VT_n\}$ - the virtualized tasks queue; $S_{optimal-low}$; $S_{optimal-high}$; T_{under}; $V_{leading-cluster}$; $V_{wing-first-cluster}$; $V_{wing-second-cluster}$; $V_{wing-third-cluster}$

Output: Self-organizing and consolidation actions

begin
  foreach $TICK$ **do**
    if $(S \in V_{leading-cluster})$ **then**
      Task_Migration$(S, V_{leading-cluster}, T_{under}, S_{optimal-high})$
      NotifyAgents$(V_{leading-cluster}, Migration)$
      if (**Distance**$(S, S_{optimal-high}) < T_{under}$) **then**
        Move$(S, V_{wing-first-cluster})$
      end if
      if (**Distance**$(S, S_{optimal-low}) < T_{under}$) **then**
        Move(**Turn_Off_Server**$(S), V_{wing-second-cluster}$)
      end if
      if $(VTqueue \neq \emptyset)$ **then**
        $VT =$ **Dequeue**$(VTqueue)$
        if (**Task_Deployment**$(S, S_{optimal-high}, T_{under}, VT)$)**then**
          NotifyAgents$(V_{leading-cluster}, Deployment)$
          if (**Distance**$(S, S_{optimal-high}) < T_{under}$) **then**
            Move$(S, V_{wing-first-cluster})$
            NotifyAgents$(V_{wing-third-cluster}, IdleToActive)$
          end if
        end if
      end if
    end if
    if $(S \in V_{wing-first-cluster})$ **then**
      if(**Distance**$(S, S_{optimal-low}) < T_{under}$) **then**
        Move(**Turn_Off_Server**$(S), V_{wing-second-cluster}$)
      end if
    end if
    if $(S \in V_{wing-second-cluster})$ **then**
      if (**ReceivedNotification**$(TurnOFFToIdle)$) **then**
        Move(**Turn_ON_Server**$(S), V_{wing-third-cluster}$)

```
      end if
    end if
    if (S ∈ V_wing−third−cluster) then
      if (ReceivedNotification(IdleToActive)) then
        Move(S, V_leading−cluster)
        NotifyAgents(V_wing−second−cluster, TurnOFFToIdle)
      end if
    end if
  end foreach
end
```

On every tick, if the server is part of the $V_{leading-cluster}$ cluster, its attached agent will try to consolidate the server workload by using task migration (lines 5-7). If after task migration a $V_{leading-cluster}$ of servers becomes fully loaded, its attached agent will decide to move it to the $V_{wing-first-cluster}$ (lines 8-10). Otherwise, if a server becomes empty, its attached agent decides to turn it off and moves it to the $V_{wing-second-cluster}$ (lines 11-13).

As long as there is an incoming workload, the attached agents of the $V_{leading-cluster}$ will try to properly deploy the virtualized tasks on the cluster of servers. If a server from $V_{leading-cluster}$ can accommodate the current virtual task, then the task is deployed and the attached agent signals the task deployment to the attached agents of $V_{leading-cluster}$ (see lines 14-17). After deployment, the server load level is tested and if it is fully loaded then it is shifted to the $V_{wing-first-cluster}$ cluster and a notification that a server must be moved from idle state to active state is sent to the agents attached to $V_{wing-third-cluster}$ servers (see lines 18-21). Following this notification, an agent attached to a server part of the $V_{wing-third-cluster}$ wakes up a server form idle state (the server is randomly chosen) and moves it to the $V_{leading-cluster}$ to accommodate the incoming workload (lines 35-40). At the same time, a notification is sent to the $V_{wing-second-cluster}$ to signal that a turned-off server must be powered-on and put in idle state (line 38) to replace the server that was moved in the $V_{leading-cluster}$.

If the server is part of the $V_{wing-first-cluster}$, its attached agent periodically evaluates if the server deployed workload has finished its execution (see line 26). If this is the case, the server is turned off and moved to the $V_{wing-second-cluster}$ containing turned off servers (see line 27). For the $V_{wing-second-cluster}$, if a server attached agent receives a turn off to idle notification (see line 31) the server is turned on and moved to $V_{wing-third-cluster}$ (line 32).

To identify the appropriate servers from the V-formation leading cluster on which a task can be *deployed* the following steps are taken: (i) a representation of the servers after virtualized tasks deployment is constructed (see Algorithm 3.4, line 5) and (ii) the degree of similarity between the configuration of the servers after deployment and the optimal one is calculated (see Algorithm 3.4, line 6).

[!h] Task Deployment Algorithm **Input:** S; $S_{optimal-high}$; T_{under}; $VT_j = [CPU_{reqj}, MEM_{reqj}, HDD_{reqj}]$
Output: Boolean value signalling VT_j correct deployment

```
begin
  bool = false
  S_new = Server_Load_After_Deploy(S, VT_j)
  d = Distance(S_new, S_optimal−high)
  if (d < T_under) then
    bool = Deploy(VT_j, S)
  end if
  return bool
end
```

To calculate the degree of similarity of server configurations the following formula is used:

$$d(S_i, S_j) = [|CPU_{S_i} - CPU_{S_j}|, |MEM_{S_i} - MEM_{S_j}|, |HDD_{S_i} - HDD_{S_j}|] \tag{3.11}$$

The server, for which the distance is the smallest (all three term inequalities should be satisfied), is selected to accommodate the task (see Algorithm 3.4 lines 7-9).

For migrating virtualized tasks, the data centre servers are analysed to identify the servers that can accommodate virtual tasks from the server subject of migration. Firstly the algorithm tests if the server computing resources are inefficiently utilized (see Algorithm 3.4, line 5). This situation appears mainly because the workload virtualized tasks have different execution times and consequently, some of them will last longer than others. If the distance between the server computational resources utilization values and the optimal configuration values ($S_{optimal-high}$) drops below a specific value (for example 35%) then the virtualized tasks deployed on the server are iterated for migration (see Algorithm 3.4, lines 6-12).

[!h] Task Migration Algorithm **Input:** S; $V_{leading-cluster}$; T_{under}; $S_{optimal-high}$;
Output: Virtual tasks are migrated
begin
  $bool = false$
  if (**Distance**$(S, S_{optimal-high}) < [35\%, 35\%, 35\%]$) **then**
    foreach VT_j *deployed on* S **do**
      foreach $S_i \neq S \in V_{leading-cluster}$ **do**
        if (!$bool$) **then**
          $bool =$ **Task_Deployment**$(S_i, S_{optimal-high}, T_{under}, VT_j)$
        end if
      end foreach
    end foreach
  end if
end

4. Case Study and Results. This section presents and discusses the results obtained by using the proposed swarm-inspired self-organizing and consolidation technique.

4.1. Test Case Description. Due to costs, management and security constraints, it is very difficult to deploy and test the proposed solution in a large, real data centre. Therefore, for evaluating the swarm inspired data centre self-organizing technique, a data centre with 2000 servers was simulated. Each simulated server models the characteristics of a real server and is represented as an object having as main attributes the CPU, memory and hard disk capabilities. The real server used as a model for the simulated server has the following hardware configuration: CPU - Intel(R) i7 870 2.93GHz, memory - 6GB DDR3 and hard disk - 750GB. The servers attached agents which implement the data centre self-organizing and consolidation algorithm are developed using the JADE framework [37].

To test the power saving capabilities of the swarm-inspired technique, a random workload that needs to be accommodated in the simulated data centre has been generated. The workload consists of groups of virtual tasks with different requirements, arriving sequentially, during a time interval of one hour. Each virtual task is described by its request for server resources, as defined in Formula 3.1.

It is assumed that the data centre is managed by OpenNebula, the state of the art middleware for managing virtualized data centre clouds [33]. OpenNebula uses a server consolidation algorithm based on Fit First approach [34] (which places the incoming workload on the first server in which it will fit). Two scenarios are considered for the test case: (1) the workload is accommodated in the simulated data centre by the default OpenNebula Fit First consolidation algorithm and (2) the proposed self-organizing and consolidating technique is in charge, replacing the OpenNebula algorithm.

For the first scenario, the Fit First server consolidation algorithm which deploys the new coming tasks on the first server found that can accommodate it, also overprovisions the computing resources to handle peak load values. In this case, there will always be servers in the data centre that are powered-on without executing workload.

In the second scenario, using the proposed data centre self-organizing and consolidating technique, the number of up and running servers that accommodate the workload tasks is variable. If the size of the workload increases over the total capacity of the $V_{leading-cluster}$, then the intelligent agents collaborate to adjust the number of servers in the cluster in order to be able to distribute and deploy all of the workload virtual tasks. If after the task deployment, there are servers in the $V_{leading-cluster}$ which don't execute any workload, the attached agents decide to power off these servers to save energy.

To assess the energy consumption improvement, a one hour workload that needs to be deployed in the data centre has been generated. In the evaluation process two aspects are considered for both scenarios when measuring the energy efficiency of the self-organizing and consolidating technique: (i) computing the Green Performance Indicator (GPI), DH-UR and (ii) estimating the average power consumption.

4.2. DH-UR Metric Evaluation. The Deployed Hardware Utilization Ratio (DH-UR) [26] is a Green Performance Indicator which measures the number of IT computing resources (servers) that consume energy without doing any actual work (see Formula 4.1). In other words, it measures the energy required by the idle systems, or the amount of wasted energy.

$$DH-UR = \frac{Number\ of\ Servers\ Running\ Applications}{Number\ of\ Servers\ Up\ and\ Running} \tag{4.1}$$

The value of this metric for a green data centre should be close to 1, meaning that the up and running servers are actually processing tasks. The DH-UR values for the simulated data centre have been calculated for both consolidation algorithms. Table 4.1 shows the average DH-UR values highlighting the fact that the proposed swarm-inspired approach improves the utilization of server resources and at the same time optimizes the energy efficiency. It can be noticed that the Fit First approach keeps servers under-utilized leading to a lower average value for DH-UR indicator.

TABLE 4.1
Average DH-UR metric values for the two scenarios

Algorithm	Average DH-UR Value
Swarm-inspired data centre self-organizing and consolidating algorithm	0.98
Fit First server consolidation algorithm	0.73

Figure 4.1 presents the DH-UR value evolution for the two test cases. It can be noticed that when using the swarm-inspired technique, the DH-UR value is close to 1 for all test period, while for the OpenNebula default Fit First algorithm it barely jumps over 0.8. The latter value reflects the case of today's real data centres.

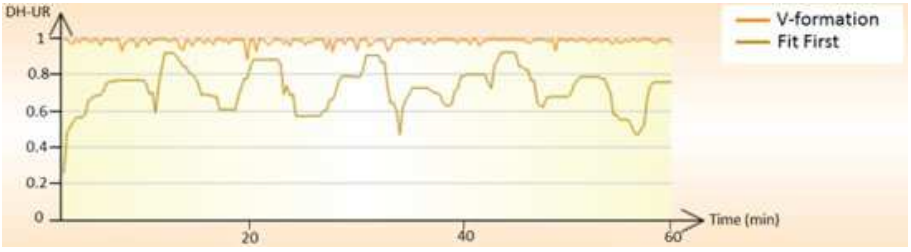

FIG. 4.1. *DH-UR Green Indicator for the two test cases*

4.3. Power Consumption Evaluation. For power consumption estimation, the same simulated data centre with 2000 homogenous servers has been used. Each simulated server models the characteristics of a real server: CPU - Intel(R) i7 870 2.93GHz, memory - 6GB DDR3 and hard disk - 750GB.

To estimate the data centre power efficiency, the instant power consumption of the real server has been measured using a power meter, in three situations: (i) the server is up and running and does not execute any tasks (Idle Power Mode), (ii) the server is up and running and executes workload (Working Power Mode) and (iii) the server is up and running and fully loaded (Full Load Power Mode). The server power consumption was measured using the ISO-TECH IPM3005 [35] power meter and the obtained results are listed in Table 4.2.

TABLE 4.2
The data centre server states and their associated instant power consumption

Power Mode	Load Level	Instant Power Consumption
Idle	0-10%	70W
Working	10-80%	100W
Full Load	80-100%	130W

For estimating the whole data centre power demand, the following metric has been defined and used:

$$\begin{aligned} PW(DC) \;=\; & \#IdleServers * PowerIdle + \\ & \#WorkingServers * PowerWorking + \\ & \#FullLoadServers * PowerFullLoad + \\ & PW_{overhead} \end{aligned} \tag{4.2}$$

where $\#IdleServers$ is the number of data centre servers that are in Idle Power Mode, $\#WorkingServers$ represents the number of servers that are in Working Power Mode, $\#FullLoadServers$ is the number of servers in Full Load Power Mode, while $PW_{overhead}$ is the power overhead induced by the algorithms' management operations.

$PW_{overhead}$ is estimated by measuring the instant power consumption of each management action defined by the swarm inspired technique. The following types of management actions are measured: (1) deploy workload virtual task on a leading cluster server, (2) migrate a task between two servers from the $V_{leading-cluster}$, (3) hibernate server when completing its workload execution and (4) wake up a server to accommodate incoming workload.

The measured overhead for each action (together with the software tools used for enforcing the actions) is listed in Table 4.3. Apart from the OpenNebula middleware, SSH and WakeOnLan software tools are used to turn off and on the servers. The movement actions between different V-formation clusters are logical actions and do not consume any extra energy.

TABLE 4.3
Power consumption overhead of the swarm-inspired technique actions

Action	Involved Tools	Instant Power Consumption
Deploy	OpenNebula	10W
Migrate	OpenNebula	20W
Turn off Server	SSH	100W
Wake up Server	WakeOnLan	100W

The overall power consumption overhead of the swarm-inspired technique is estimated as follows:

$$\begin{aligned} PW_{overhead} \;=\; & \#TasksDeployed * PowerDeploy + \\ & \#TasksMigrated * PowerMigrate + \\ & \#ServerTurnedOn * PowerWakeUp + \\ & \#ServerTurnedOff * PowerTurnOff \end{aligned} \tag{4.3}$$

where $\#TasksDeployed$ represents the number of workload tasks that were deployed, $\#TasksMigrated$ is the number of migrated workload tasks, $\#ServerTurnedOn$ is the number of servers that were turned on, while $\#ServerTurnedOff$ represents the number of turned off servers.

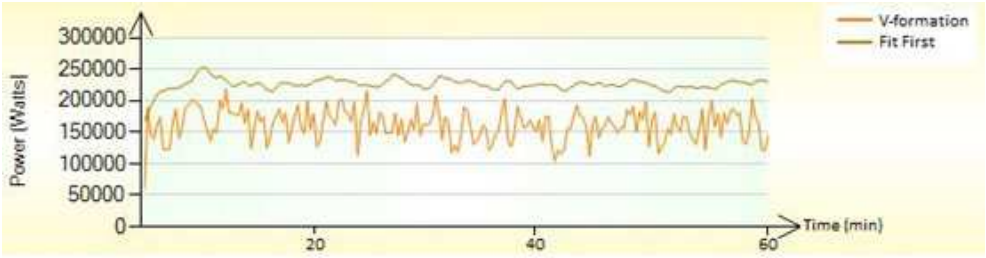

FIG. 4.2. *Power consumption evolution for the two test cases*

Figure 4.2 shows the variation of the estimated power consumption for the simulated data centre when accommodating a randomly generated workload with the two consolidation approaches alternatively activated.

When using the proposed swarm-inspired self-organizing and consolidating approach, only the servers that execute workload ($V_{leading-cluster}$ and $V_{wing-first-cluster}$) are up and running and consuming energy . The number of $V_{wing-first-cluster}$ idle servers is low and a high number of servers are kept in turned off state in the $V_{wing-third-cluster}$. The higher power demand of the Fit First consolidation algorithm is due to the high number of idle servers that are not executing tasks while demanding 70 Watts instant power.

The calculated average power demand values for the two tested scenarios are given in Table 4.4. The average values were computed for 2000 servers in the data centre and for one hour time interval.

TABLE 4.4
Average power consumption

Algorithm	Average Power Demand
Swarm-inspired data centre self-organizing and consolidating algorithm	161 670 W
Fit First server consolidation algorithm	226 322 W

As a result, an average power saving of around 40% has been estimated for the swarm-inspired approach compared with the OpenNebula default Fit First consolidation algorithm. Even though this percentage is really high it should be noted that the data centre IT facility aspect was neglected during simulation. In a real data centre, the IT facilities usually demand about 60% of the total [28]. This means that the power saving of 40% of the proposed solution represents about 16% from the total power demand of a data centre.

5. Conclusions. In this paper a swarm-inspired data centre consolidation technique which aims at increasing the data centre computing resources utilization ratio and implicitly its power efficiency is proposed. The results are promising showing that using the swarm-inspired technique the data centre Deployed Hardware Utilization Ratio increases with about 34%. The average power saving is around 40% from the power consumed by the data centre computing resources and about 16% from its total power consumption including the IT facility.

Acknowledgments. This work has been partially supported by the GAMES project [36] and has been partly funded by the European Commission IST activity of the 7th Framework Program (contract number ICT-248514). This work expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this work.

REFERENCES

[1] S. MURUGESAN, *Harnessing Green IT: Principles and Practices*, IEEE IT Professional, (2008) pp. 24-33.
[2] N. E. JERGER, D. VANTREASE, AND M. LIPASTI, *An Evaluation of Server Consolidation Workloads for Multi-Core Designs*, Proceedings of the IEEE International Symposium on Workload Characterization, (2007), pp. 47-56.

[3] M. Marzolla, O. Babaoglu and F. Panzieri, *Server Consolidation in Clouds through Gossiping*, Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, (2011), pp. 1-6.
[4] B. W. Verdaasdonk, H. Koopman and F. van der Helm, *Energy efficient walking with central pattern generators: from passive dynamic walking to biologically inspired control*, Biological Cybernetics, Springer-Verlag, Volume 101, Issue 1 (2011), pp. 49-61.
[5] P. Boonma and J. Suzuki, *Biologically-inspired Adaptive Power Management for Wireless Sensor Networks*, Handbook of Wireless Mesh & Sensor Networking, Chapter 3.4.8, McGraw-Hill (2008), pp. 190 - 202.
[6] P. Champrasert and J. Suzuki, *SymbioticSphere: A Biologically-Inspired Autonomic Architecture for Self-Adaptive and Self-Healing Server Farms*, Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, (2006), pp. 469-474.
[7] D. Barbagallo, E. Di Nitto, D. J. Dubois and R. Mirandola, *A Bio-Inspired Algorithm for Energy Optimization in a Self-organizing Data Centre*, Self-Organizing Architectures - First International Workshop, SOAR 2009, Cambridge, UK, September 14, 2009, Revised Selected and Invited Papers, LNCS, Volume 6090, (2010), pp. 127-151.
[8] C. Lee, H. Wada and J. Suzuki, *Towards a Biologically-inspired Architecture for Self-regulatory and Evolvable Network Applications*, Advances in Biologically Inspired Information Systems: Models, Methods and Tools, Studies in Computational Intelligence Volume 69, (2007), pp. 21-45.
[9] J. Torres, et al., *Tailoring Resources: The Energy Efficient Consolidation Strategy Goes Beyond Virtualization*, International Conference on Autonomic Computing, (2008), pp. 197 - 198.
[10] Y. Ajiro and A. Tanaka, *Improving Packing Algorithms for Server Consolidation, Proceedings of the Computer Measurement Group's*, (2007).
[11] J. Berral, et al., *Towards energy-aware scheduling in data centres using machine learning*, Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, (2010) pp. 215-224.
[12] L. Wang, et al., *Towards Thermal Aware Workload Scheduling in a Data Centre*, Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks, (2009) pp. 116-122.
[13] E. Kalyvianaki, and T. Charalambous, *On Dynamic Resource Provisioning for Consolidated Servers in Virtualized Data Centres*, Proceedings of the 8th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS-8), (2007).
[14] B. Speitkamp and M. Bichler, *A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centres*, IEEE Transactions on services computing, Volume 3, Issue 4, (2010).
[15] O. Niehorster and A. Brinkmann, *Autonomic Resource Management Handling Delayed Configuration Effects* Third IEEE International Conference on Coud Computing Technology and Science, pp. 138 - 145, 2011.
[16] J. O. Kephart, et al., *Coordinating Multiple Autonomic Managers to Achieve Specified Power-Performance Tradeoffs*, International Conference on Autonomic Computing, (2007).
[17] T. Cioara, I. Anghel, I. Salomie, G. Copil, D. Moldovan and B. Pernici, *A context aware self-adapting algorithm for managing the energy efficiency of IT service centres*, Ubiquitous Computing and Communication Journal, Volume 6 No. 1, (2011).
[18] L. Barroso, *The Price of Performance*, ACM Queue, Volume 3, Issue 7, (2005), pp. 48-53.
[19] R. J. Anderson, E. W. Mayr and M. Warmuth, *Parallel Approximation Algorithms for Bin Packing*, Stanford University, 1988.
[20] M. Poniatowski, *Foundations Of Green IT: Consolidation, Virtualization, Efficiency, and ROI in the Data Centre*, Prentice Hall, 2009.
[21] G. H. Schueller, and S. K. Schueller, *Animal Migration (Animal Behaviour)*, Chelsea House Pub (Library), 2009.
[22] B. Davis and K. Davis, *Marvels of Creation: Breathtaking Birds*, Master Books, 2006.
[23] C. L. Henderson, *Birds in Flight - The Art and Science of How Birds Fly*, Voyageur Press, 2008.
[24] P. Berthold, *Bird Migration - A General Survey*, Second Edition - Oxford University Press, 2001.
[25] J. M. Kaplan, W. Forrest and N. Kindler, *Revoluzioning Data Centre Energy Efficiency*, McKinsey&Company, 2008. Available online at: http://www.mckinsey.com/clientservice/bto/pointofview/pdf/Revolutionizing_Data_Centre_Efficiency.pdf
[26] J. Stanley, K. Brill and J. Koomey, *Four Metrics Define Data Centre Greenness*, Uptime Institute, white paper, Available online at: http://uptimeinstitute.org.
[27] S. Srikantaiah, A. Kansal and F. Zhao, *Energy Aware Consolidation for Cloud Computing*, Microsoft Research, 2009. Available online at: http://research.microsoft.com/pubs/75408/srikantaiah_hotpower08.pdf
[28] Climate Savers Computing Initiative, http://www.thegreengrid.org/Home/about-the-green-grid/TGGCSCI.aspx
[29] K. Brill, *Data Centre Energy Efficiency and Productivity*, The Uptime Institute, Whitepaper, 2007.
[30] C. B. Pop, I. Anghel, T. Cioara, I. Salomie, I. Vartic, *A Swarm-inspired Data Center Consolidation Methodology*, International Conference on Web Intelligence, Mining and Semantics, Article No. 41, 2012.
[31] U.S. Environmental Protection Agency, ENERGY STAR Program, Report to Congress on Server and Data Centre Energy Efficiency, Public Law 109-431, 2007.
[32] SMART 2020: Enabling the low carbon economy in the information age, Report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI), 2008.
[33] OpenNebula, the open source toolkit for cloud computing, http://opennebula.org/.
[34] OpenNebula scheduler, http://opennebula.org/documentation:archives:rel3.0:schg
[35] ISO-TECH IPM3005, http://www.iso-techonline.com/products/iso-tech-electrical-installation-testers.html
[36] GAMES FP7 Research Project, http://www.green-datacentres.eu/.
[37] Jade, Java Agent Development Framework, http://jade.tilab.com.

[38] The Green Grid, http://www.thegreengrid.org/

Edited by: Viorel Negru and Daniela Zaharie
Received: May 24, 2013
Accepted: Jul 9, 2013

DOI 10.12694/scpe.v14i2.842
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. 83–94. http://www.scpe.org

MULTIPLE EQUIVALENT SIMULTANEOUS OFFERS STRATEGY IN AN AGENT-BASED GRID RESOURCE BROKERING SYSTEM – INITIAL CONSIDERATIONS

NAOUAL ATTAOUI[*] MARIA GANZHA[†] MARCIN PAPRZYCKI[†] KATARZYNA WASIELEWSKA[†] AND MOHAMMAD ESSAAIDI [‡]

Abstract. The *Agent in Grid* (*AiG*) project attempts to integrate the concept of the Grid and an agent-based system to facilitate efficient resource management in the Grid. In this paper, we present preliminary considerations concerning multiple equivalent simultaneous offers strategy that can be used in the *Service Level Agreement* (*SLA*) negotiations. These negotiations are the key part of the main use case scenarios within the *AiG* project. In this context, first, we describe the *AiG* system. Second, we introduce the simultaneous offer strategy, as a mechanism known from economy, and suggest an approach for using it in the *AiG* negotiations.

Key words: Grid computing, multiple equivalent simultaneous offers, AiG, agent-based negotiations, SLA, multi-attribute negotiations

1. Introduction. The context for the presented work is provided by the *Agents in Grid* project (*AiG*; [16, 17, 23]) that aims at creating a management system for the Grid computing resources. The system utilizes the concept of software agents that can act flexibly in a dynamic and uncertain environment, where resources can appear and disappear at any time. Furthermore, the *AiG* project integrates business principles, by allowing users to earn money by offering resources, or to pay for the use of available resources. The key part of the realization of this economic model is the negotiation process that should lead to contracts beneficial to the engaged parties. In the *AiG* project, negotiations materialize in the two main scenarios: (i) when the user would like to execute her job and is searching for resources that satisfy her needs, (ii) when a resource owner is seeking a team to join, to earn money. Here, we are confronted with one of the key assumptions behind the *AiG* project. Based on the analysis of working conditions of volunteer systems (e.g. seti@home; [32]), and taking into account progress in the development of agent systems (their capability for autonomously reaching an agreement [33]) it was stipulated that agents that offer their resources to earn money should work in teams. In this way, teams should be able to deliver results even if some members disappear without warning (for more details, see [16]).

Observe that negotiations taking place in the *AiG* project have to take into account multiple attributes. In an example given in [23] simple job execution negotiations involved simultaneous evaluation of *price*, *job start time* and *job end time*. Currently, after the *AiG* ontology was fully developed (see, [22] for more details), truly multi-criterial negotiations can be used. Note that in both negotiation scenarios introduced above, the negotiation strategy is constrained by the following conditions: (1) no information about the preferences of the other participant(s) in the negotiation, and (2) the agreement reached should be optimal to both negotiating parties.

Let us observe that recent studies in social psychology and economics [1, 2] have shown that using a strategy based on multiple offers is more profitable than a strategy based on a single offer. Here, the proposer can choose combinations of parameter values in such a way to obtain equivalent offers i.e. they generate equal values of the proposer's utility function. On the other hand, the second party in negotiations is more likely to be satisfied with one of these offers. Taking this into account, in this paper we would like to consider how the multi-offer mechanism can be introduced into the *AiG* system.

The remainder of the paper is organized as follows. Section 2 briefly describes the *AiG* system. Next, in section 3, the related work concerning multi-offer negotiations is reviewed. We follow, in section 4, with a description how to generate multiple equivalent offers in the *AiG*, and suggests how to model this strategy in the context of the *AiG* system. We conclude with a summary and description of future work.

2. System overview.

[*]Information and Telecommunication Systems Laboratory, Faculty of Science, Tetuan, Morocco
[†]Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
[‡]National School of Computer Science and Systems Analysis, ENSIAS, Rabat, Morocco

2.1. *AiG* infrastructure. Let us start from a high-level overview of the *AiG* system. The *AiG* approach is based on (i) software agents that facilitate resource management in the Grid [15], and (ii) use of ontology for knowledge representation, and semantic data processing for knowledge utilization. As mentioned before, the *AiG* system constitutes a "return" to the roots of Grid computing, where providers were to sell their resources in the Grid, while their customers were to pay for the resource use. In other words, the *AiG* project attempts at (re)integrating business mechanisms into the Grid. This being the case, in the system we distinguish two roles, which are interchangeable: resource (service) provider and client (resource consumer). Finally, the key part of the proposed approach is: combining agent into teams. As stated above, this is a result of an analysis of realistic situations that take place in the *global Grid*. Here, the global Grid is contrasted with the *local Grid*, where resources run under control of one or more administrators. Let us now summarize main assumptions that have been driving the development of the agent aspect of the system:

(i) Users and workers in the system are represented by *LAgents*.
(ii) Agents work in teams [16, 17].
(iii) Each team is managed by its leader, the *LMaster* agent.
(iv) *Client Information Center* (represented by the *CIC* agent) plays the role of a central repository, where information about all teams and their member workers is stored [25].
(v) In addition to the *LMaster* one of the worker agents plays a role of *LMirror*. This agent stores a copy of the information necessary for the team to persist in the case when the *LMaster* crashes.

The proposed approach can be thus represented in the form of a Use Case Diagram depicted in Fig. 2.1.

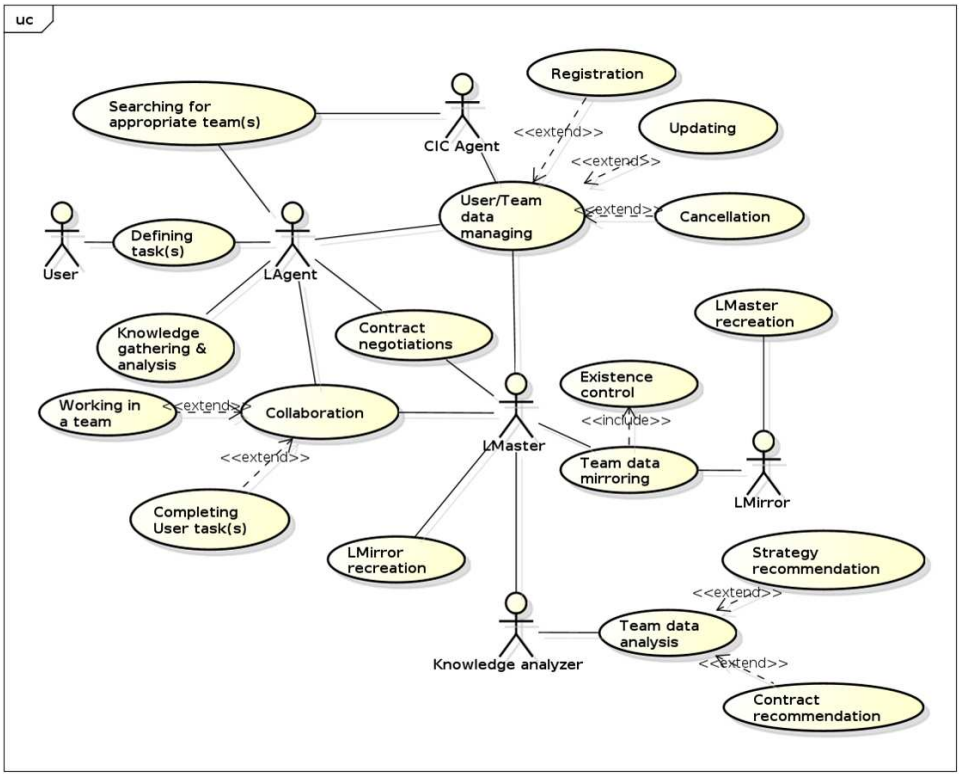

FIG. 2.1. Use Case Diagram of *AiG* system.

The work of the system is structured around two main scenarios: a user who wants to execute a job, and a user who tries to join a team. In the first scenario, when the user is seeking a team to execute her job, she specifies job constraints via web application that is ontology-driven and transforms the constraints into an ontology fragment. Next, using a special *Gateway Agent*, the interface forwards this fragment to the *LAgent*. The *LAgent* contacts the *CIC* to obtain the list of teams that can execute the job (under the defined constraints).

Next, the Iterated FIPA Contract Net Protocol [26] is applied to negotiate the *Service Level Agreement* (*SLA*). The second, team joining, scenario is very similar. The only difference is in the constraints passed by the *LAgent* to the *CIC*. Here, the *CIC* is matching the description of the potential worker against requirements of teams that would like to have more members. In the context of this paper, the crucial observation is that the Iterated FIPA Contract Net Protocol can be modified to support the multi-offer strategy. Recall that the negotiation process covers multiple parameters and decisions about joining a team, accepting new workers, choosing a team to execute a job, or accepting to execute a job involve multi-criteria analysis (see section 2.3).

2.2. Ontologies in the system. The representation of the knowledge with ontologies has recently attracted attention, as a promising approach to development of a new generation of information systems (including the Semantic Web). Furthermore, agreeing with J. Hendler, who in [14] argued that software agents and ontologies should work very well together; it was decided that in the *AiG* system all concepts will be ontologically represented. Thus, the envisaged ontology had to encompass the structure of the Grid, the characteristics of the resources, and the negotiations that occur in both scenarios.

Preliminary research in this area involved analyzing existing Grid ontologies (see [18]). On the basis of this analysis it was concluded that the closest and the most adequate ontology, for the *AiG* system, was the *Core Grid Ontology* (*CGO*; [19, 20]). However, some modifications and extensions had to be brought to this solution, especially so that it would include concepts related to the *SLA*. The complete description of the resulting ontology can be found in [21, 22]. Here, let us briefly outline its main features. For better modularity, and thus maintainability, we have split the constructed ontology into three (sub)ontologies:

1. *AiG Grid Ontology*, which is an extension of the original *CGO* that adds additional classes and properties that make specifying the hardware and software configuration of a Grid resource easier and more complete.
2. *AiG Conditions Ontology*, which contains concepts necessary to describe the terms of contracts between entities that are involved in business relationships: (a) the *LAgent* representing a user offering its resources and the *LMaster* looking for workers, and (b) the *LAgent* looking for a team to execute a job and the *LMaster* representing teams offering to run this job.
3. *AiG Messages Ontology* is used internally by the proposed solution and contains definitions of messages that can be exchanged between agents.

2.3. Negotiations in the system. Human negotiation is a daily act; we negotiate at work, at home, in the political domain, as well as in trading. Generally, stakeholders negotiate in order to reach an agreement, which satisfies their respective interests. Automated negotiations have the same goal, but they use software tools to assist negotiators to express their preferences, to accelerate reaching an agreement (even in complex situations [35]), and to help making a good decision. In our case, we must facilitate negotiations in the two scenarios mentioned earlier: (i) a user has some requirements concerning resource(s) needed to execute a job, while the resource owner wishes to maximize her gain, (ii) a prospective worker who wants to "sell" a resource is negotiating with a team that would like to assure an appropriate contract with the new worker. To reach agreement automated negotiations of the *SLA* between both parties are utilized. Based on earlier work [21], our ontology allows to consider, among others, the following parameters in the first scenario: (i) job execution start and end time, (ii) penalty for delay of job execution, and (iii) price. Parameters that can be negotiated in second scenario are (this list is, again, not exhaustive) : (i) revenue, (ii) contract duration, (iii) possibility to extend contract, and (iv) guaranteed utilization. Obviously, we deal with multi-attribute negotiations that can generate complications on several levels: generating offers, evaluating proposals, guessing negotiator preferences, etc.

From the above discussion, it can be concluded that autonomous negotiations, and the *SLA* management, are a key part of the *AiG* system (see [23] for more details). To be sure that the negotiation reaches its goal, we should choose protocols and strategies appropriate for the situation. Therefore, after considering the European projects integrating business and Grid infrastructure [24], it was decided to base the negotiations on the FIPA Iterated Contract-Net Protocol [26], to give the possibility for the user to modify his offer if no team presented an acceptable offer. Here, negotiations can involve both negotiable, e.g. deadline penalty, job execution timeline, and static parameters, e.g. resource description provided by the user.

Fig. 2.2 represents the sequence diagram of a situation when a user is attempting to execute a job (with

a help of her *LAgent*). The *LAgent* registers with the *CIC*, if not registered already. Afterwards, it requests from the *CIC* a list of teams that have workers representing resources meeting the criteria. As a result the *CIC* agent responds with a list of *LMaster* agents representing suitable teams. After that, the *LAgent* negotiates directly with the *LMaster*s. In the initial step, the *LAgent* sends out the *Call-For-Proposal (CFP)* message to all *LMaster*s of teams acquired from the *CIC*. On the basis of the *CFP*, *LMaster*s prepare their responses. Responses from the *LMaster*s can have the following forms: (1) refusal (an ACL *Refuse* message), or (2) a specific offer (an ACL *Propose* message). The *LAgent* awaits a predefined time for responses and then finds the best of them using a multi-criteria analysis (MCA). The team that has the best offer is selected to execute the job. If no acceptable offer is received, a revised *CFP* is sent to as selected subset of participants, while others are removed from further considerations.

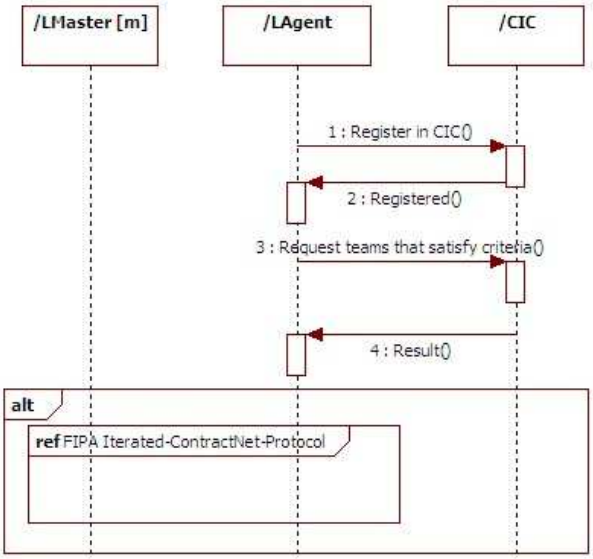

FIG. 2.2. Sequence diagram for job execution scenario.

In this paper we consider possibility of further developing/extending the negotiation mechanism through application of the Multiple Equivalent Simultaneous Offers strategy, which is introduced in the next section.

3. Multiple Equivalent Simultaneous Offers strategy. Multiple Equivalent Simultaneous Offers (MESOs) [1] is a technique used in negotiations where a set of attributes is involved. The principle is to make multiple offers at once. One negotiating party should make offers having the same utility and different values for individual parameters. In this case, the client will have more chance to be satisfied by an offer and thus negotiators are more likely to reach an agreement.

Recent research in social psychology and economy suggests that the choice of negotiation strategy based on the simultaneous offers is more profitable than the unique offer. A number of scholars have conducted studies and published articles related to the MESOs. Authors in [1] have demonstrated numerous benefits (distributive, integrative, and interpersonal) of making MESOs by conducting four online experiments, in which participants were offered a single offer, or two offers. Here, authors believe that the use of MESOs allows negotiators to get more information about each other across subsequent negotiation rounds and to achieve better overall outcomes. Other research ([3]), shows that negotiators who use MESOs achieve better outcomes than those who make a single packaged offer, without sacrificing relationships or losing credibility. While there is not a fixed (best overall) number of offers, authors of [2] suggest to present no more than three offers at a time. Similarly, according to [4], three equivalent offers can be a good strategy. Here, authors describe a software company that, in financial negotiations, presented three equivalent software packages to its clients at once: for example, a $1

million package with payment in 30 days, the same software for $1.5 million with payment in 120 days, and an enhanced package for $1.35 million with a 30-day payment. This strategy received a positive response from the customers, and profits of the company rose. Yinping Y et. al. confirmed in [5] the effectiveness of a novel strategy based on simultaneous equivalent offers to make a significant positive impact on both the economic and social-psychological outcomes of negotiations. Authors of [28] opt for multiple offers to decrease the number of negotiation rounds. They exposed the difficulties of generating multiple equivalent offers. To solve the problem, they started from the idea of finding equivalent offers in the sense that they have the same utility value. The solution was to change the values of various attributes and adjust the price to balance the offer to the desired utility value. Finally, O. Kivanc has shown in [29] that when MESO is applied, the customer's utility increases significantly without decreasing the buyer's utility. To generate multiple offers in the automated negotiations, he has utilized graph-based statistical clustering to partition the space of the offers .

Based on these considerations, it can be stipulated that in an environment of an open business Grid, like that considered in the *AiG* project, simultaneous equivalent offers could increase the chances to reach success in negotiations between team leaders and a new worker (team joining scenario), or between team leaders and a user that tries to contract a job execution (job executing scenario). In fact, a user or a new worker, will more likely be satisfied by one of the offers from the set proposed by the teams, which can reduce the number of rounds of negotiations.

Therefore, in what follows, we suggest how the *AiG* system can be modified to use simultaneous equivalent offers, instead of the basic single offer iterative strategy.

4. Multiple Equivalent Simultaneous Offers strategy in the *AiG* system.

4.1. Generating multiple equivalent offers in the *AiG*. The *AiG* approach uses autonomous *SLA* negotiations, facilitated by agents representing Grid resource providers and consumers. Using the MESO strategy in the project makes us ponder the necessary changes in the team joining and the job execution scenarios. Currently, when the *LAgent* sends a *CFP* to a list of teams, each one of them answers with a single offer. However, when the MESO approach is to be used, each response would contain a set of offers (Fig. 4.1) covering the space of possible deals acceptable to the given team. Note that this would, among others, require modification the ontology so that one message could contain several offers.

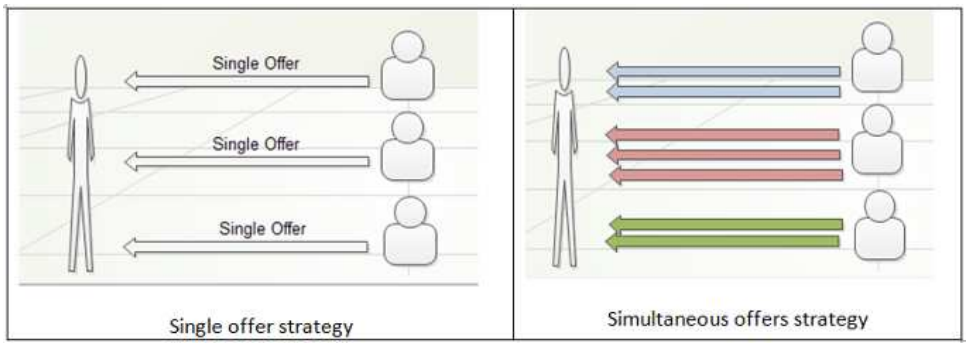

FIG. 4.1. Single offer strategy and simultaneous offers strategy.

Let us now see in details how offers can be prepared in an example of the job execution scenario: user is looking for a team to commission job to be executed on a resource that should run Windows Vista SP2 operating system. He would like to meet the deadline 20 of September 2013, with a penalty of at least 10 monetary units for the delay of contract completion. Furthermore, the maximum price that can be charged for the job is 100 monetary units. First, our user would specify resource requirements using the ontology-driven GUI (see, [34]). These requirements would be forwarded to the *LAgent* that would communicate with the *CIC* to find teams that satisfy them. The user would also provide its *LAgent* with contract conditions that would be used in negotiations. In such negotiations, first, a *CFP* message is sent by the *LAgent* to the *LMaster*s of the selected teams. Note that all information exchanged in the system (including requirements and contract

offer) is ontologically demarcated. In this example, we use three negotiation parameters: deadline penalty, job execution time line, and payment conditions with three properties: (i) delay penalty, (ii) fixed utilization price, and (iii) lead time. For each of these parameters the user specifies its importance by assigning weight that are later used in the MCA.

The following snippet shows declaration of selected properties, classes and a simple class expression with restrictions on the contract, where the required resource should run Windows Vista SP2 operating system, the deadline penalty should be at least 10 , the fixed resource utilization price less than 100 and the execution job should meet the deadline of 20 of September 2013:

```
<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#contractedResource -->

<owl:ObjectProperty
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#contractedResource">
  <rdfs:range rdf:resource="&cgo;GridComponent"/>
  <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description
    rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#JobExecutionConditions"/>
    <rdf:Description
    rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#WorkerContractConditions"/>
   </owl:unionOf>
  </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#fixedUtilizationPrice -->

<owl:ObjectProperty
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#fixedUtilizationPrice">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:comment rdf:datatype="&xsd;string">
  A price for the availability of a resource paid irrespective of offered/used ServiceChargableItems.
  </rdfs:comment>
  <rdfs:domain
   rdf:resource="http://gridagents.sourceforge.net/AiGConditionsOntology#PaymentConditions"/>
  <rdfs:range rdf:resource="http://gridagents.sourceforge.net/AiGConditionsOntology#Pricing"/>
</owl:ObjectProperty>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#jobExecutionTimeline -->

<owl:ObjectProperty
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#jobExecutionTimeline">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain
   rdf:resource="http://gridagents.sourceforge.net/AiGConditionsOntology#JobExecutionConditions"/>
  <rdfs:range rdf:resource="&time;TemporalEntity"/>
 </owl:ObjectProperty>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#paymentConditions -->

<owl:ObjectProperty
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#paymentConditions">
 <rdf:type rdf:resource="&owl;FunctionalProperty"/>
 <rdfs:range
rdf:resource="http://gridagents.sourceforge.net/AiGConditionsOntology#PaymentConditions"/>
 <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description
    rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#JobExecutionConditions"/>
    <rdf:Description
    rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#WorkerContractConditions"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
</owl:ObjectProperty>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#deadlinePenalty -->

<owl:DatatypeProperty
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#deadlinePenalty">
 <rdf:type rdf:resource="&owl;FunctionalProperty"/>
 <rdfs:domain
rdf:resource="http://gridagents.sourceforge.net/AiGConditionsOntology#JobExecutionConditions"/>
 <rdfs:range rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#offPeakTimePrice -->

<owl:DatatypeProperty
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#offPeakTimePrice">
 <rdfs:domain
rdf:resource="http://gridagents.sourceforge.net/AiGConditionsOntology#Pricing"/>
 <rdfs:range rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#JobExecutionConditions -->

<owl:Class
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#JobExecutionConditions"/>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#PaymentConditions -->
```

```
<owl:Class
rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#PaymentConditions"/>

<!-- http://gridagents.sourceforge.net/AiGConditionsOntology#Pricing -->

<owl:Class rdf:about="http://gridagents.sourceforge.net/AiGConditionsOntology#Pricing"/>

<!-- http://www.owl-ontologies.com/unnamed.owl#GridComponent -->

<owl:Class rdf:about="&cgo;GridComponent"/>

<!-- http://www.w3.org/2006/time#TemporalEntity -->

<owl:Class rdf:about="&time;TemporalEntity"/>
```

The *CFP* is equivalent to:

```
JobExecutionConditions that contractedResources some
(WorkerNode that isRunningOS value vista_sp2) and paymentConditions some
(PaymentConditions that fixedUtilizationPrice some (Pricing that peakTimePrice some float[<= 100]))
and deadlinePenalty some float[< 10] and jobExecutionTimeline some dateTime[=2013-09-20T23:59:59]
```

In response to such *CFP*, the *LMaster*s may respond with job execution contract offers. Contract offer presented below specifies the fixed utilization price to be 100, the deadline penalty to be 10 and the job deadline to be 20 of September 2013. In what follows, we represent this offer as: $Offer1 = (100, 10, 20/09/2013)$.

```
<owl:NamedIndividual rdf:ID="Contract">
    <deadlinePenalty rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >10.0</deadlinePenalty>
        <rdf:type rdf:resource="#JobExecutionConditions"/>
    <jobExecutionTimeline>
      <owl:NamedIndividual rdf:ID="Sept20">
        <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
        >2013-09-20T23:59:59</time:inXSDDateTime>
        <rdf:type rdf:resource="http://www.w3.org/2006/time#Instant"/>
      </owl:NamedIndividual>
    </jobExecutionTimeline>
    <paymentConditions>
      <PaymentConditions rdf:ID="PaymentConditions_1">
        <fixedUtilizationPrice>
          <Pricing rdf:ID="Pricing_2">
            <offPeakTimePrice rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
            >100.0</offPeakTimePrice>
          </Pricing>
        </fixedUtilizationPrice>
      </PaymentConditions>
    </paymentConditions>
  </owl:NamedIndividual>
```

As mentioned before, in the multiple-offer approach there is more than one offer to be made by each team. However, it is assumed all of them have the same utility. Thus, we have to evaluate $Offer1$, which combines multiple attributes. According to the multi-attribute utility theory (MAUT; [30]), each team has a (private) utility function $(U_p(v), U_d(v), U_j(v))$ for each criterion (*fixedUtilizationPrice*, *deadlinePenalty* and *jobExecutionTimeline* respectively). The utility function expresses the team's evaluation of each value in the criterion's domain.

To define the attribute's utility function we refer to [31]. For each attribute, the team chooses a minimum (a), a maximum (c) and a best (b) values. On their basis we can distinguish three cases:

(i) In many situations, when the minimum is considered the best, appropriate equation is as follows:

$$\left[U(v) = \begin{cases} crl1, v \leq a \\ (v-c)/(a-c), a < v < c \\ 0, v \geq c \end{cases}\right] \tag{4.1}$$

(ii) When the maximum value is considered the best, the utility function can be defined as:

$$\left[U(v) = \begin{cases} crl0, v \leq a \\ (c-v)/(c-a), a < v < c \\ 1, v \geq c \end{cases}\right]$$

(iii) Finally, when the best utility is placed in between the minimum and the maximum, the following equation is considered to be the most adequate:

$$\left[U(v) = \begin{cases} crl(v-a)/(b-a), a \leq v \leq b \\ (v-c)/(b-c), b \leq v \leq c \\ 0, Otherwise \end{cases}\right]$$

Now we must look at the overall offer. In our example, weight is assigned to each criteria, and W_p, W_d and W_j are the weights for *fixedUtilizationPrice*, *deadlinePenalty* and *jobExecutionTimeline*, respectively. Note here that those weights are different from those of the user; they concern teams' assessment of importance of each criterion. Here, the method of assigning weights is out of scope of this paper.

The combination of attribute values in the offer gives a particular utility value according to the utility formula expressed as follows:

$$Utility(offer1) = U_p(100) \times W_p + U_d(10) \times W_d + U_j(20/09/2013) \times W_j \tag{4.2}$$

Recall that offers equivalent to the previous ones (but with a different attribute combination), should have the same utility value for the team:

$$Utility(offer1) = Utility(offer2) \tag{4.3}$$

To solve this problem, we have decided to use the method proposed in [28]. The idea is to change the values of various attributes and to balance to the same utility value of the first offer by adjusting the cost.

In our case we suggest to apply random values dR for the *deadlinePenalty* and jR for *jobExecutionTimeline*. Thus we have:

$$Utility(offer2) = U_p(x) \times W_p + U_d(dR) \times W_d + U_j(jR) \times W_j \tag{4.4}$$

where x is the value of the price that we have to look for, in order to satisfy equation (4.3).

Then we deduce that:

$$U_p(x) = \frac{Utility(offer1) - U_d(dR) \times W_d - U_j(jR) \times W_j}{W_p} \tag{4.5}$$

Therefore, generating another offer means finding a price that brings to the team the utility expressed in equation (4.5).

We can generate additional offers by adopting the same method. Note that this method is also suitable for the team joining scenario, for instance by considering in the offer the following attributes: *contract Period*, *Availability*, *guaranteed utilization* and *payment*.

The next section presents messages exchanged in the *AiG* negotiations process that implement the above approach. These messages will be exchanged in the context of a modified FIPA Iterated contract net protocol.

4.2. MESOs modeling. Following the guidelines found in [1], we fix the number of offers generated by teams to no more than three.

There are two possibilities of handling the MESO in the *AiG*, depending on the possibility to assemble offers in a single message. Note that we initially illustrate the different approaches using a simplified version of the protocol – FIPA Contract Net Protocol. Only later, we present the adopted solution in the context of the Iterated FIPA Contract Net Protocol.

4.2.1. Single *Propose* message including multiple offers. Here, a *Propose* message can contain more than one offer, and each *LMaster* sends one *Propose* message with a list of offers included. The *Accept-Proposal* sent to the *LMaster* of a chosen team contains reference to the accepted offer (Fig. 4.2).

4.2.2. A separate *Propose* message for each offer. In the second approach, each *Propose* message contains a single offer. The *LMaster* of the selected team receives an *Accept-Proposal* message for the accepted offer and the *Refuse-Proposal* messages for the remaining offers (Fig. 4.3).

4.2.3. Proposed solution. To generate a smaller number of messages, and because handling a set of offers jointly is more clear conceptually and easier practically, we opt for the first solution, where all offers are sent in a single message. Here, the response of the *LAgent*, will be unique and will use the already implemented solution, with the only difference that it will specify (in the *Accept-Proposal* message) the offer that was selected, among the received ones. However, as a result, we need to change our ontology, by adding an identifier to the

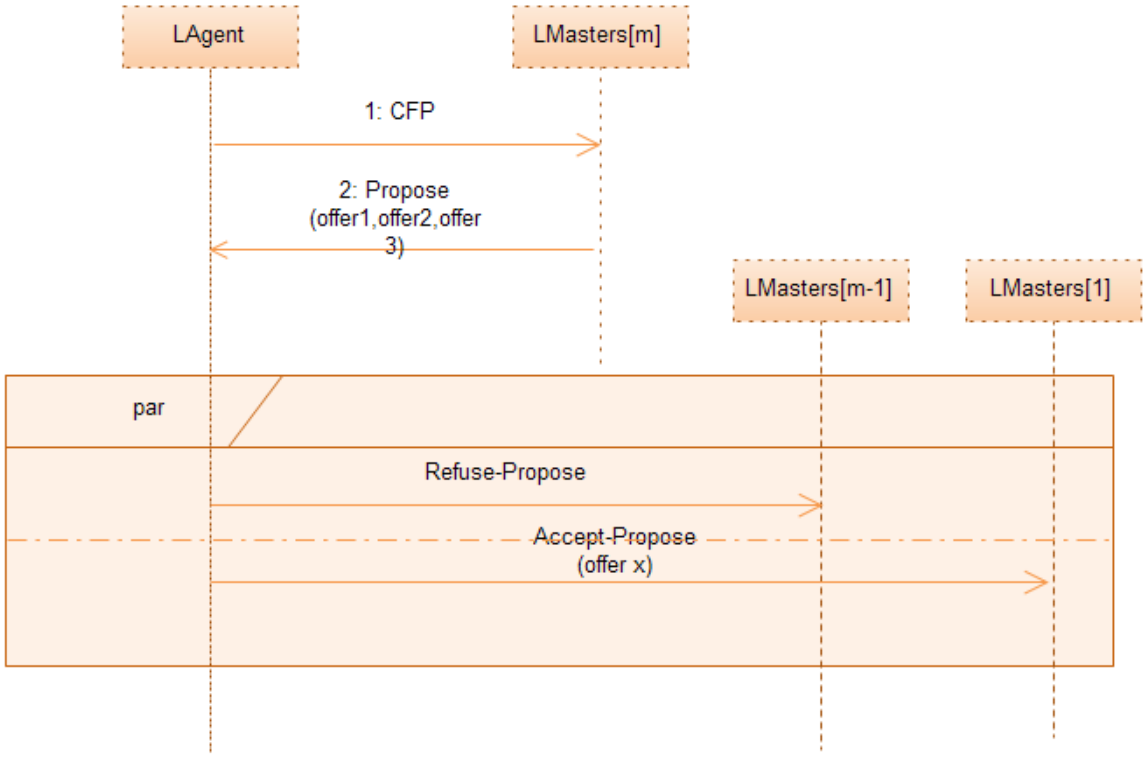

FIG. 4.2. One *Propose* with all offers.

offer accepted for the contract (in the *Accept-Proposal* message). The *Propose* message has also to be changed, to allow it to contain a list of offers.

Fig. 4.4 presents a sequence diagram for the modified Iterated FIPA Contract Net Protocol that is to be utilized in the *AiG* project to support multiple offers. The modified messages exchanged in the team joining negotiation scenario, which are very similar to the job execution scenario, are as follows:

1. Teams submit to the *CIC* characteristics of their potential workers.
2. The *LAgent*, representing the user, sends an ACL message of type *CFP*, that contains the description of worker's resource (resource that it is willing to offer e.g. available software, operating system, processor type, etc.) and constraints concerning contract terms, such as the contract length.
3. A message of type *Propose* sent by the *LMaster* containing details of the multiple (though no more than three) contracts offers it is proposing to the *LAgent*.

If it's the last round:

(i) Messages of type *Accept-Proposal* and *Refuse-Proposal* are sent respectively to the *LMaster* of the selected offer (specifying, which offer was accepted), and to the remaining *LMaster* agents,

(ii) *Inform-Done* message sent by the *LMaster* that received the *Accept-Proposal*.

If this is not the last round, revised *CFP* is send to a selected subset of *LMasters*, and rejections to the others.

5. Concluding remarks. Multiple equivalent simultaneous offers is a novel strategy that begins to play an important role in the field of economics. A project that will include an economic aspect, such as the *AiG*, should consider using this technique, in order to benefit from its advantages.

The goal of this paper was to discuss how to integrate the MESOs in a multi-attribute negotiation within an agent-based Grid resource management system. Thus, we had to consider generating a package of offers in an environment where no information about negotiator preferences is available. Since this is the first attempt

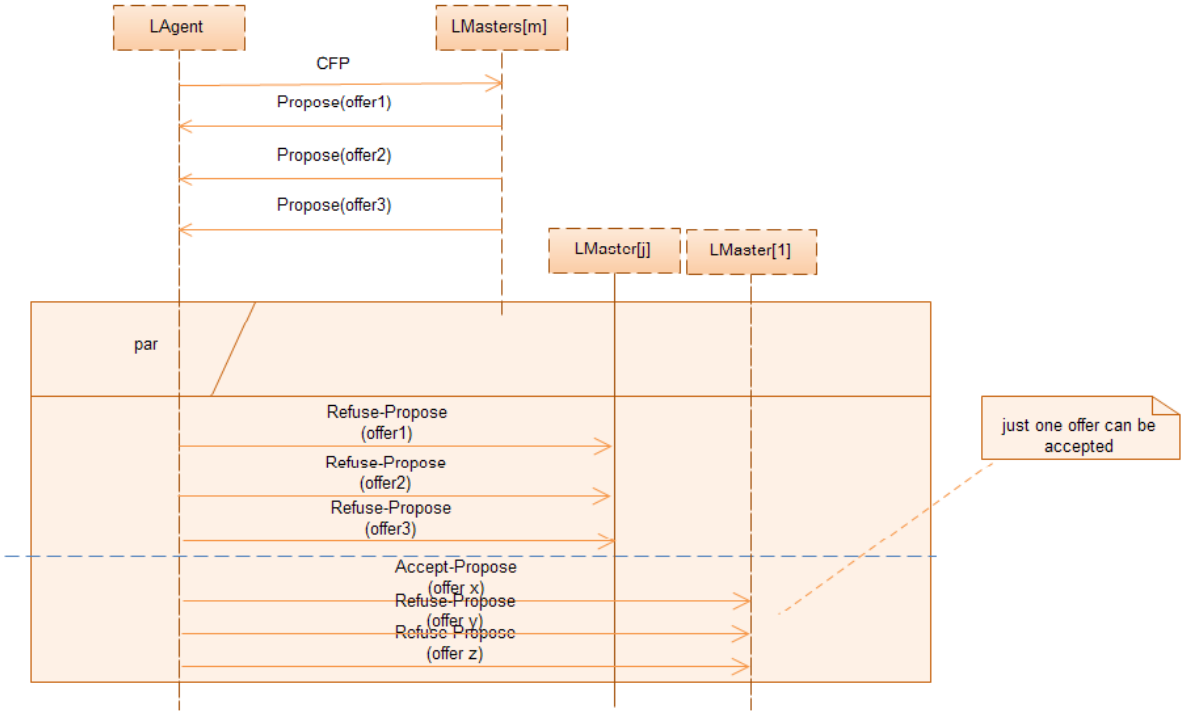

FIG. 4.3. a *Propose* message and a response for each offer.

at integrating the multiple equivalent offers strategy into negotiations in the *AiG* project, there are still some adaptations to be made, especially to the ontology. Also, we should note that this strategy requires a complex preparations prior to the negotiation, and a question that remains open is, will it be always possible to generate multiple offers. Filling these gaps and implementing the approach are our next research goals. The implementation will help us to compare MESO with single strategy and to evaluate experimentally what happens the number of simultaneous offers is changing.

REFERENCES

[1] V. H.MEDVEC, L.GEOFFREY, A. D.GALINSKY, AND C. S.ALETHA, *Choice and Achievement at the Bargaining Table: The Distributive, Integrative, and Interpersonal Advantages of Making Multiple Equivalent Simultaneous Offers*, IACM 18th Annual Conference, June 2005. Available at SSRN: http://ssrn.com/abstract=732665 or http://dx.doi.org/10.2139/ssrn.732665.

[2] MANDELBAUM AND ROBB, *How to Negotiate Effectively*, INC Magazine. http://www.inc.com/magazine/20101101/how-to-negotiate-effectively.html. Retrieved 27 September 2012.

[3] V. H. MEDVEC AND A. D. GALINSKY, *Putting More on the Table: How Making Multiple Offers Can Increase the Final Value of the Deal*, Harvard Business Review. http://hbr.org/product/putting-more-on-the-table-how-making-multiple-offe/an/N0504B-PDF-ENG.

[4] V. H. MEDVEC AND A. D. GALINSKY, *The Value of Making Several Offers in Business Negotiations*, http://www.pon.harvard.edu/daily/business-negotiations/the-value-of-making-several-offers-in-financial-negotiations/. Retrieved 27 September 2012.

[5] Y. YINGPING, S. SHARAD AND X. YUNJIE (CALVIN),*Offer with Choices and Accept with Delay: A Win-Win Strategy Model for Agent Based Automated Negotiation*,(2009). ICIS 2009 Proceedings. Paper 180. http://aisel.aisnet.org/icis2009/180.

[6] D. OUELHADJ, J. GARIBALDI, J. MACLAREN, R. SAKELLARIOU, AND K. KRISHNAKUMAR, *A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing*, In Proceedings of the 2005 European Grid Computing Conference (EGC 2005), February 2005, p. 651–660.

[7] M. PARKIN, P. HASSELMEYER, B. KOLLER AND P. WIEDER, *An SLA Re-negotiation Protocol*, Proceedings of the 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC08), CEUR Workshop Proceedings, ISSN 1613-0073, Volume 411. Dublin, Ireland. November 2008.

[8] A. ANDRIEUX, *Web Services Agreement Specification (WS-Agreement)*, Proposed Recommendation, Open Grid Forum,

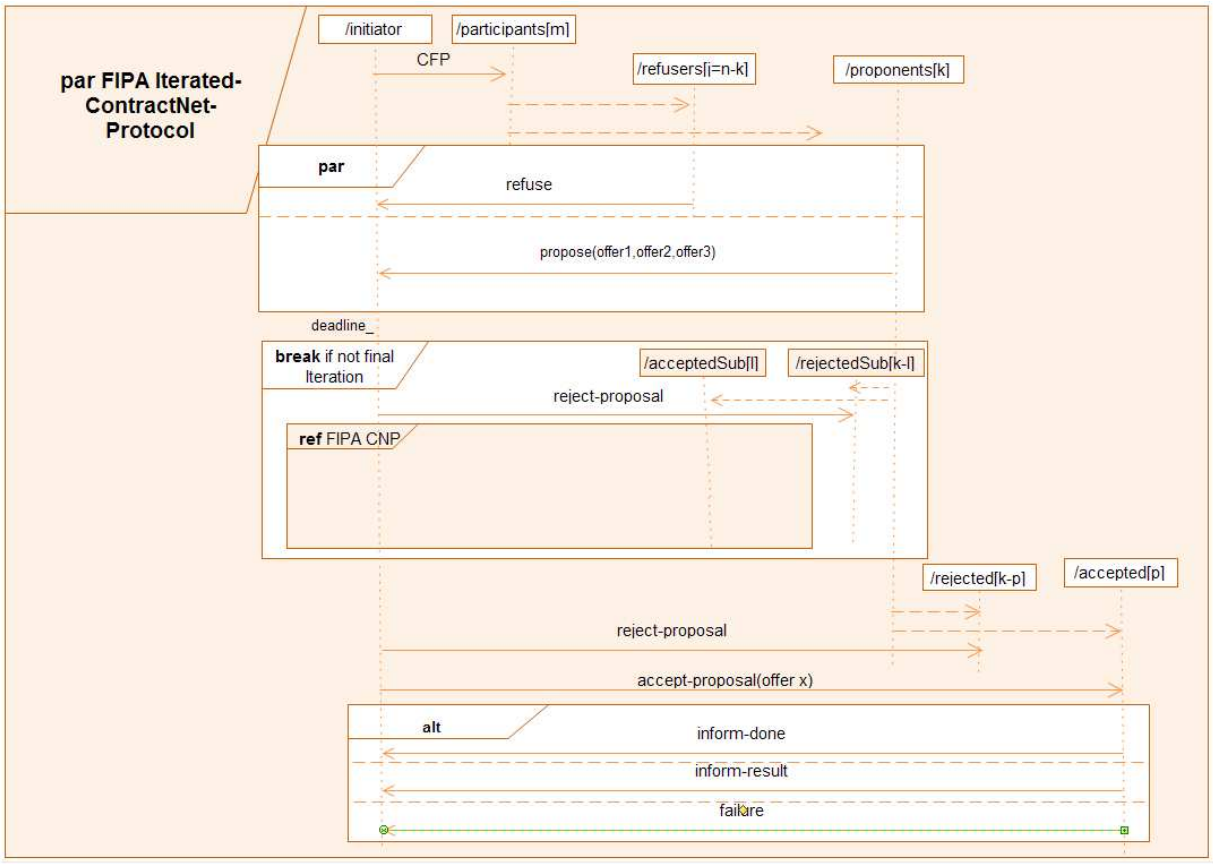

FIG. 4.4. Modified FIPA Iterated Contract Net Protocol.

September 2006. Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG).

[9] G. D. MODICA, V. REGALBUTO, O. TOMARCHIO AND L.VITA, *Enabling re-negotiations of SLA by extending the WS-Agreement specification*, IEEE International Conference on Services Computing (SCC 2007), pp. 248–251.

[10] CoreGRID European Research Network on Foundations, software infrastructures and applications for large scale distributed, GRID and peer to peer technologies, http://www.coregrid.net/

[11] NextGRID: Architecture for next generation Grids, http://www.nextgrid.org/

[12] BREIN Business objective driven Reliable and Intelligent Grids for Real BussiNess, http://www.eu brein.com/

[13] Grid Resource Allocation Agreement WG (GRAAP WG), http://www.ogf.org/gf/group info/view.php?group=graap wg arbitrary

[14] J. HENDLER, *Agents and the semantic web*, IEEE Intelligent Systems, 16(2), pp. 30-37.

[15] I. FOSTER, N. R. JENNINGS AND C. KESSELMAN, *Brain meets brawn: Why grid and agents need each other*, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Vol. 1, 2004, pp. 8–15.

[16] W. KURANOWSKI, M. GANZHA, M. GAWINECKI, M. PAPRZYCKI, I. LIRKOV, AND S. MARGENOV, *Forming and managing agent teams acting as resource brokers in the grid-preliminary considerations*, International Journal of Computational Intelligence Research, Vol. 4, No. 1, 2008, pp. 9–16.

[17] W. KURANOWSKI, M. GANZHA, M. PAPRZYCKI AND I. LIRKOV, *Supervising agent team an agent-based grid resource brokering system-initial solution*, Proceedings of the Conference on Complex, Intelligent and Software Intensive Systems, 4-7 March 2008, pp. 321–326.

[18] M. DROZDOWICZ, M. GANZHA, M. PAPRZYCKI, R. OLEJNIK, I. LIRKOV, P. TELEGIN, M. SENOBARI, *Ontologies, Agents and the Grid: An Overview*,in B.H.V. Topping, P. Ivnyi, (Editors), "Parallel, Distributed and Grid Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK, Chapter 7, 2009, pp. 117–140.

[19] Core grid ontology. http://grid.ucy.ac.cy/grisen/cgo.owl.

[20] W. XING, M. D. DIKAIAKOS, R. SAKELLARIOU, S. ORLANDO, AND D. LAFORENZA, *Design and Development of a Core Grid Ontology*, In Proc. of the CoreGRID Workshop: Integrated research in Grid Computing, 2005, pp. 21–31.

[21] M. DROZDOWICZ, M. GANZHA, M. PAPRZYCKI, K. WASIELEWSKA, I. LIRKOV, R. OLEJNIK, N. ATTAOUI, *Utilization of Modified Core GRID Ontology in an Agent-based Grid Resource Management System*, PROCEEDINGS OF THE CATA 2010 CONFERENCE.

[22] M. Drozdowicz, K. Wasielewska, M. Ganzha, M. Paprzycki, N. Attaoui, I. Lirkov, R. Olejnik, D. Petcu and C. Badica , *Chapter Ontology for Contract Negotiations in Agent-based Grid Resource Management System*, in P. Ivnyi, B.H.V Topping,(Editors), Trends in parallel, distributed, grid and cloud computing for engineering, Saxe-Coburg Publications, Stirlingshire, UK, 2011.

[23] K. Wasielewska, M. Drozdowicz, M. Ganzha, M. Paprzycki, N. Attaoui, D. Petcu, C. Badica, R. Olejnik, I.Lirkov, *Negotiations in an Agent-based Grid Resource Brokering Systems*, in P. Ivnyi, B.H.V Topping,(Editors), "Trends in in Parallel, Distributed, Grid and Cloud Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK, 2011.

[24] M. Parkin, R. Badia, J. Martrat, *A Comparison of SLA Use in Six of the European Commissions FP6 Projects*, Institute on Resource Management and Scheduling, CoreGRID-Network of Excellence, April 2008, pp. 1–38.

[25] M. Dominiak, W. Kuranowski, M. Gawinecki, M. Ganzha, M. Paprzycki, *Efficient Matchmaking in an Agent-based Grid Resource Brokering System*, Proceedings of the International Multiconference on Computer Science and Information Technology, 2006, pp. 327-335.

[26] FIPA Iterated Contract Net Interaction Protocol Specification, http://www.fipa.org/specs/fipa00030/index.html

[27] FIPA Contract Net Protocol Specification, http://www.fipa.org/specs/fipa00029/SC00029H.html.

[28] Neubert, R. Gorlitz, O. Teich and Tobias, *Automated negotiations of supply contracts for flexible production networks*, International Journal of Production Economics, Elsevier, vol. 89(2), pp. 175–187, May 2004.

[29] K. Ozonat, *Multiple Offer Strategy for Automated Negotiation Agents*, http://world-comp.org/p2011/ICA3527.pdf

[30] M. Barbuceanu and w. Lo *A Multi-Attribute Utility Theoretic Negotiation Architecture for Electronic Commerce*, Proc. 4th Int. Conf. on Autonomous Agents, Barcelona, Catalonia, Spain (2000).

[31] A. D. Youngblood and T. R. Collins, *Addressing balanced scorecard trade-off issues between performance metrics using multi attribute utility theory*, Engineering Management Journal, Mar 2003; vol. 15, N1; pp. 11–17.

[32] SETI@home website. http://setiathome.ssl.berkeley.edu

[33] C. Bartolini, C. Preist, and N. R. Jennings, *A software framework for automated negotiation*, in Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications, Springer Verlag 2005, LNCS 3390: pp. 213–235.

[34] M. Drozdowicz, M. Ganzha, M. Paprzycki, P. Szmeja, and K. Wasielewska, *OntoPlay - A Flexible User-Interface for Ontology-based Systems*. AT, volume 918 of CEUR Workshop Proceedings CEUR-WS.org, (2012), pp. 86–100.

[35] C. Badica, S. Ilie, M. Kamermans, G. Pavlin, A. Penders, M. Scafes, *Multi-Agent Systems, Ontologies and Negotiation for Dynamic Service Composition in Multi-Organizational Environmental Management*. NATO Science for Peace and Security Series - D: Information and Communication Security, Volume 32: Software Agents, Agent Systems and Their Applications, (2012), pp. 286–306. http://dx.doi.org/10.3233/978-1-60750-818-2-286

Edited by: Viorel Negru and Daniela Zaharie
Received: June 1, 2013
Accepted: Jul 9, 2013

DOI 10.12694/scpe.v14i2.843
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. 95–109. http://www.scpe.org

AN AGENT-BASED APPROACH FOR HYBRID MULTI-CLOUD APPLICATIONS

DJAMEL BENMERZOUG*

Abstract. Cloud service offerings provide a competitive advantages to enterprises through flexible and scalable access to computing resources. With the recent advances in Cloud computing, the need is emerging for interoperability between Cloud services so that a complex and developed business applications on Clouds are interoperable. In fact, combining different independent Cloud services necessitates a uniform description format that facilitates the design, customization, and composition. In this context, Agent Interaction Protocols (IP) are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts. The challenge here is twofold. First, we must propose a formal model that is rich enough to capture interactions characteristics. Second, we must allow designers to combine existing protocols to achieve a new specific need.

The work presented in this paper is considered as a first step toward Agent Interaction Protocols as a Service. In fact, we propose a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols. Also, it proposes a set of operators that allows the creation of new value-added protocols using existing ones as building blocks.

Key words: Agent Interaction Protocols, Cloud Computing, Protocols Composition, Petri Net, Verification.

1. Introduction. Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with a minimal management effort or service provider interaction [34]. According to the intended access methods and availability of Cloud computing environments, four major types of Cloud deployments are known: public Clouds, private Clouds, community Clouds, and hybrid Clouds [31].

The increasing adoption rate of Cloud computing is currently driving developers, integrators and hosting enterprises to take Cloud computing into account. As a result, enterprises need to revise their current assets as Cloud computing is becoming a strategic asset.

However, enterprises are driven by different reasons to maintain their own data center, such as legislation of storing data in-house, investments in the current infrastructure, or the extra latency and performance requirements. This drive is supported by the fact that enterprises have already invested heavily in their own private server equipment and software [21].

Consequently, we believe that a hybrid approach makes more sense for enterprises. This approach allows one to use the internal infrastructure combined with public Cloud resources to build a hybrid Cloud. For example, on one hand, critical applications can run on the infrastructure/platform of the private data center, and, on the other hand, the public Cloud can be used as a solution to manage peak demands or for disaster recovery.

While the different Cloud solutions offer support for applications development and deploying, there are different issues that require special attention through specialized developments, like resource management, service integration, or service orchestration [16].

Combining different independent Cloud services is the ability to integrate multiple services into higher-level applications. This integration necessitates a uniform description format that facilitates the design, customization, and composition. In this context, Agent Interaction Protocols (IP) are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts.

In previous work [7] [8], we described the use of interaction protocols to define and manage collaborative processes in B2B relationships where the autonomy of participants is preserved. We demonstrated the practicability of our approach by embedding it in a Web services language for specifying business protocols, which conducive to reuse, refinement and aggregation of our business protocols. We also elaborated translation rules from interaction protocols notations used in our approach into Colored Petri Nets (CPN). These rules are implemented in IP2CPN: the tool we developed to automatically generate Petri nets from protocols specifications. Resulting Petri nets can be analyzed by dedicated tools to detect errors as early as possible.

*Department of Software Technologies and Information Systems, Faculty of New Technologies of Information and Communication, University Constantine 2, Algeria. (benmerzoug@gmail.com).

Cloud services composition and orchestration have seen increased attention in published works, and researchers have introduced numerous approaches to automate it. As such, current techniques suggest that enterprises will acquire related tools and perform integration activities locally. The knowledge and expertise to perform composition activities is hard to attain and equally difficult to maintain. Thus, the knowledge obtained during Cloud services composition is not stored, reused, or shared. To be competitive, enterprises must be able to transfer and reuse knowledge attained after each composition scenario.

Driven by the motivation of reuse, we would like to treat protocols as modular components, potentially composed into additional protocols, and applied in a variety of business processes. By maintaining repositories of commonly used, generic, and modular protocols, we can facilitate the reuse of a variety of well-defined, well-understood and validated protocols. For example, a payment protocol can be used in a process for purchasing goods as well as in a process for registering for classes at a university. Further, the repository would expand as newly composed protocols are inserted into it.

In this paper, we propose a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols. Our approach provides a set of operators that allows the creation of new value-added protocols using existing ones as building blocks. Sequence, alternative, iteration, and parallel are typical constructs specified in the control flow. We also give a formal semantics to the proposed operators in terms of Petri nets.

The remainder of the paper is organized as follows: Section 2 overviews our previous work. Based on that, in Section 3, we define the concept of the Cloud business protocols. Section 4 gives details about protocols composition including protocols contract and protocols operators. Section 5 focuses on the verification of composite protocol correctness. Finally, Section 6 overviews some related work and Section 7 provides some concluding remarks.

2. Interaction Protocols in support of Service-Based Enterprise Application. From an abstract point of view of systems modelling, i.e. once we abstract from the nature of the languages and platforms over which services are deployed, the main challenge raised by service-based enterprise application is in the number of autonomic entities involved and the complexity of the interactions within them [37]. That is, the complexity that matters is not so much in the size of the code through which such entities are programmed but on the number, intricacy and dynamicity of the interactions in which they will be involved. This is why it is so important to put the notion of interaction at the center of research in service-based enterprise application modelling. This is also why new methods and formal techniques become necessary.

To tackle these challenges, our approach is based on the notion of IP through which we specify complex services and break the complexity of running systems by recognising larger chunks that have a meaning in the application domain. This notion of IP, which is inspired by work of Multiagent Systems (MAS), supports the modelling of composite services as entities whose business logic involves a number of interactions among more elementary service components.

Figure 2.1 shows our conceptual model for the treatment of service-based enterprise application. Business partners participating in this scenario publish and implement a public process. So, a public business process is the aggregation of the private processes and/or Web services participating in it.

Let us notice that private processes are considered as the set of processes of the enterprise itself and they are managed in an autonomous way to serve local needs. The public processes span organizational boundaries. They belong to the enterprises involved in a B2B relationship and have to be agreed and jointly managed by the partners.

Interaction protocols have been used in the area of multi-agent systems to represent interactions among agents. In the context of B2B relationships, a business protocol is a specification of the allowed interactions between two or more participant business partners. These interactions represent public business processes that enterprises agreed on the collaboration.

In our case the public process is specified by interaction protocols, which are used by the integrator agent to enact the public process at run time. Interaction protocols should be published, shared and reused among the business partners.

2.1. Specification of Interaction Protocols. The B2B integration scenarios typically involve distributed business processes that are autonomous to some degree, hence the importance of Interaction protocols

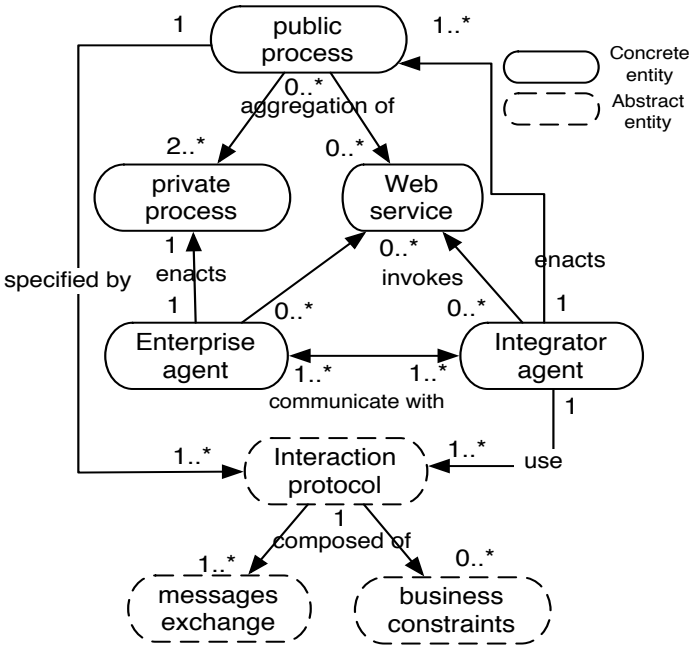

FIG. 2.1. *Conceptual Model: Enterprise Application Based on Interaction Protocols*

based modelling. They are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts. Formally an interaction protocol is defined as follow:

Definition 1. An Interaction Protocol is a quadruplet:

IP = <ID, R, M, f_M >, where:

- ID is the identifier of the interaction protocol
- $R = r_1, r_2, \ldots, r_n$ $(n > 1)$ is a set of Roles (private business process or Web Services)
- M is a set of non-empty primitive (or/and) complex messages, where:
  - A Primitive Message (PM) corresponds to the simple message, it is defined as follow:
    PM = <Sender, Receiver, CA, Option>, where:
    - Sender, Receiver ∈ R
    - CA ∈ FIPA ACL Communicative Act[1] (such as: cfp, inform, ...)
    - Option: contain additional information (Synchronous / Asynchronous message, constraints on message,...)
  - A Complex Message (CM) is built from simpler (primitive) ones by means of operators:
    $CM = PM_1 op PM_2 \ldots op PM_m$, where:
    - $m > 1$, op ∈ {XOR, OR, AND}, and
    - ∀ i ∈ [1,m-1], $PM_i.Sender = PM_{i+1}.Sender$, $PM_i.Sender \in R$.
- f_M: a flow relation defined as : $f_M \subseteq$ (RxR), where (RxR) is a Cartesian product $(r_1, r_2) \in$ (RxR), for $r_1, r_2 \in$ R

Developing effective protocols to be executed by autonomous partners is challenging. Similar to protocols in traditional systems, IP in open and Web-based settings need to be specified rigorously so that business partners can interact successfully.

For this reason, we have developed a method for IP design and verification. The proposed method (see figure 2.2) uses different models and languages.

[1]FIPA: Foundation for Intelligent Physical Agents [36].

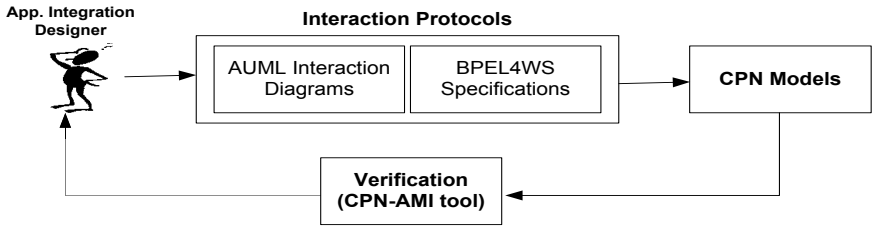

FIG. 2.2. *The proposed method*

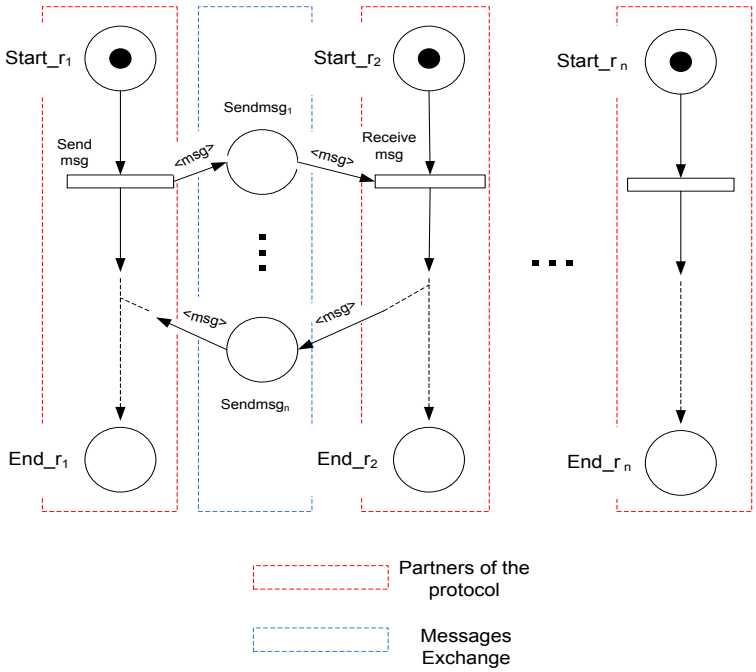

FIG. 2.3. *An Example of Interaction Protocol*

Our method motivates the use of IP based on AUML/BPEL4WS, where pre- and post-conditions, rules, guards are specified in OCL[2].

AUML (Agent UML) notation [1] [2] is a UML profile dedicated to agents trying to simplify the transition from software engineering to multi-agent system engineering. In other hand, BPEL4WS [22] (Business Process Execution Language for Web Services) is a de facto standard for describing Web services composition. In our context, BPEL4WS was used as a specification language for expressing the interaction protocols of the multi-agent system [6].

In previous work [7], we elaborated translation rules from interaction protocols notations used in our approach into Colored Petri nets. These rules are implemented in IP2CPN: the tool we developed to automatically generate the Petri net from protocols specification. The resulting Petri net specification can be analyzed by CPN-AMI tool [17] to detect errors as early as possible.

The CPN representation in our approach introduces the use of colors to represent additional information about business processes interaction states and communicative acts of the corresponding interaction. Colors sets are defined in the net declaration as follow (the syntax follows standard CPN-notation used in [20]).

[2]OCL: Object Constraint Language [28].

Color sets :
Communicative Act = **with** FIPA ACL Communicative Acts;
Role = string **with** "a" ..."z" ;
Content = string **with** "a" ..."z" ;
Bool = **with** true—false;
MSG = record
s,r: Role;
CA: Communicative Act;
C: Content;

The MSG colour set describes the communicative acts and is associated with the net message places. The $MSGs$ colored token is a record $< s, r, ca, c >$, where the s and r elements determine the sender and the receiver of the corresponding message. These elements have the color set $Role$, which is used to identify business processes or/and Web services participating in the corresponding interaction. The $CommunicativeAct$ and the $Content$ color sets represent respectively the FIPA-ACL communicative acts and the content of the corresponding message. We note that the places without color set hold an indistinguishable token and therefore have the color domain token = $\{\bullet\}$.

In our CPN representation, each role is considered equivalent to a type of resource, which is represented in a Petri net as a place. Hence, there will be one token in the place for each actor playing this role. As shown in figure 2.3, each one of these places is labelled $Start_{r_i}$ where r_i is the role name.

The life line of role is represented implicitly by a place and transitions sequence belonging to this role. The net is constituted therefore by one sub-net (Petri net process) for each role acting during the interaction and these nets are connected by places that correspond to the exchanged messages.

3. Cloud Computing and Enterprises Application Integration. Cloud computing infrastructures can allow enterprises to achieve more efficient use of their IT hardware and software investments. They do this by breaking down the physical barriers inherent in isolated systems, and automating the management of the group of systems as a single entity. Cloud computing is an example of an ultimately virtualized system, and a natural evolution for data centers that employ automated systems management, workload balancing, and virtualization technologies.

3.1. Cloud Computing Models. According to the intended access methods and availability of Cloud computing environments, there are different models of deployment [31]. They include private Cloud, public Cloud, community Cloud, and hybrid Cloud, which are briefly analyzed below (see figure 3.1).

private Cloud: In this model, the Cloud infrastructure is exclusively used by a specific organization. The Cloud may be local or remote, and managed by the enterprise itself or by a third party. There are policies for accessing Cloud services. The techniques employed to enforce such private model may be implemented by means of network management, service provider configuration, authorization and authentication technologies or a combination of these.

public Cloud: Infrastructure is made available to the public at large and can be accessed by any user that knows the service location.

community Cloud: Several organizations may share the Cloud services. These services are supported by a specific community with similar interests such as mission, security requirements and policies, or considerations about flexibility. A Cloud environment operating according to this model may exist locally or remotely and is normally managed by a commission that represents the community or by a third party.

hybrid Cloud: Involves the composition of two or more Clouds. These can be private, community or public Clouds which are linked by a proprietary technology that provides portability of data and applications among the composing Clouds.

According to the Forrester Research market [23], many businesses want interoperability between their internal infrastructure combined with public Cloud resources. They might want to use private application to process data in the Cloud, they might want to use a Cloud-based application to process private data, or they

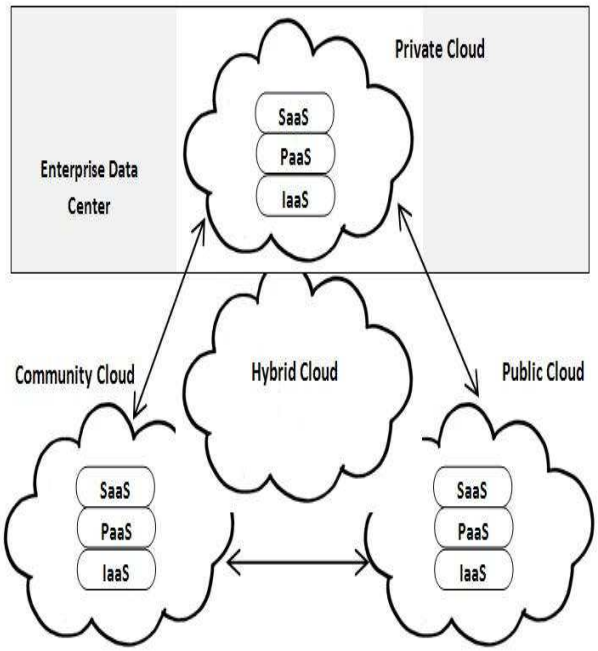

FIG. 3.1. *Cloud Computing Models*

might want to use applications or tools that will run both private and on the public Cloud. Consequently, we believe that a hybrid approach makes more sense for enterprises. In such approach, there is a need for complex developed business applications on the Clouds to be interoperable. Cloud adoption will be hampered if there is not a good way of integrating data and applications across Clouds.

In the next section, we introduce the Cloud Business Protocols, which are a useful way for structuring interaction among Cloud resources, by organising activities into relevant contexts and providing a common guide to the all parts.

3.2. The Cloud Business Protocol. As stated in [27], to ensure a meaningful composition of two or more Cloud services, there is a clear need for placing emphasis on how to develop enhanced composite service offerings at the application-level and assign or reassign different virtual and physical resources dynamically and elastically. In fact, combining different independent Cloud services necessitates a uniform description format that facilitates the design, customization, and composition.

Business protocol is a specification of the allowed interactions between two or more participant business partners. Applied to Cloud computing, we propose the following variation: *A Cloud Business Protocol is two or more business parties linked by the provision of Cloud services and related information.*

In fact, business parties in the Cloud computing area are interconnected by the Cloud business protocol. These parties are involved in the end-to-end provision of products and services from the Cloud service provider for end Cloud customers. Because protocols address different business goals, they often need to be composed to be put to good use. For example, a process for purchasing goods may involve protocols for ordering, shipping, and paying for goods.

Driven by the motivation of reuse, we would like to treat protocols as modular components, potentially composed into additional protocols, and applied in a variety of business processes. By maintaining repositories of commonly used, generic, and modular protocols, we can facilitate the reuse of a variety of well-defined, well-understood, and validated protocols.

In the rest of this paper, we propose a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions

for protocols.

4. Towards an Approach for Business Protocols Composition. In traditionally approaches, protocols are either not described formally or are described merely in terms of message order. As states by [19], such approaches give a basis for structuring communicative interaction among business partners, but do not address the fundamental knowledge engineering challenge of reuse and composition at the level of protocols.

Specifically, because protocols address different business goals, they often need to be composed to be put to good use. For example, an enterprise that is interested in selling books could focus on this protocol while outsourcing other protocols such as payment and shipment.

The composition of two or more business protocols generates a new protocol providing both the original individual behavioral logic and a new collaborative behavior for carrying out a new composite task. This means that existing protocols are able to cooperate although the cooperation was not designed in advance.

Definition 2. (Composite Protocol). A Composite Protocol (CP) is a tuple $CP = (P, Op, Input, Output, P_{init}, P_{fin})$ where:

- P is a non empty set of *basic protocols*,
- Op is a non empty set of *operators*, $Op \subseteq (P \times P) \cup (P \times CP) \cup (CP \times P) \cup (CP \times CP)$,
- $Input, Output$ are a set of the elements required (produced) by the composite protocol CP,
- P_{init} is non empty set of *initial* protocols, $P_{init} \in P$ and
- P_{fin} is non empty set of *final* protocols, $P_{fin} \in P$.

Modelling protocols composition requires control structures involving loops, choice and parallelism. This will enable complex behaviour of business protocols, such as concurrent execution, or iteration while a certain condition holds. Consequently, an automated means of assembling complex protocols from atomic ones is essential.

In this section, we introduce a theoretical approach for structural protocols composition that allows the creation of new value-added business protocols using existing ones as building blocks. Sequence, alternative, iteration, and arbitrary sequence are typical constructs specified in the control flow.

The proposed approach provides the underpinnings of aggregation abstractions for protocols. To achieve this goal, we require an agreement between business protocols in the form of a shared contract.

4.1. Protocol Contract. For a correct composition of protocols, they must come to an agreement or contract. A contract describes the details of a protocol (participants in the protocol, produced elements, required elements, constraints,...) in a way that meets the mutual understandings and expectations of two or more protocols. Introducing the contract notion gives us a mechanism that can be used to achieve a meaningful composition.

Definition 3. (Protocol Contract). A contract C, is a collection of elements that are common across two protocols. It represents the mutually agreed upon protocol schema elements that are expected and offered by the two protocols.

1. Let us denote by P_k^{in}, P_k^{out} the set of elements required (produced) by the protocol P_k where $P_k^{in} = \{x_i, i \geq 1\}$ and $P_k^{out} = \{y_j, j \geq 1\}$
2. Let us define a function θ, called a *contract-mapping*, that maps a set of P_k^{out} elements (produced by the protocol P_k) and a set of P_r^{in} (consumed by the protocol P_r), $\theta = \{\exists\, y_i \in P_k^{out} \wedge \exists\, x_j \in P_r^{in} \mid (x_i, y_j)\}$, which means that the protocol P_r consumes the element y_j provided by the protocol P_k, and $C = (P_k^{out})_\theta(C) \cup (P_r^{in})_\theta(C)$.

4.2. Protocols Compositability relationship. The compositability relationship means that business protocols can be joined together to create a new protocol that performs more sophisticated applications. That composition can then serve as a protocol itself. Figure 4.1 shows an example of business protocols composition where the composite protocol has six sub-protocols (called $P_1, P_2, \ldots, P_6$) and ⧫ is a parallel operator.

We can classify the nature of protocols composition on the dependance degree between them. We may thus distinguish between two kinds of Compositability relationship:

Definition 4. (Partial Compositability). Two protocols P_k and P_r meet the Partial Compositability relationship *iff* $\exists\, x_i \in (P_r^{in})_\theta(C), \exists\, y_j \in (P_k^{out})_\theta(C) \mid (x_i, y_j)$, which means that the protocol P_r can be executed after the protocol P_k but it must wait until all its "required elements" will be provided by others protocols.

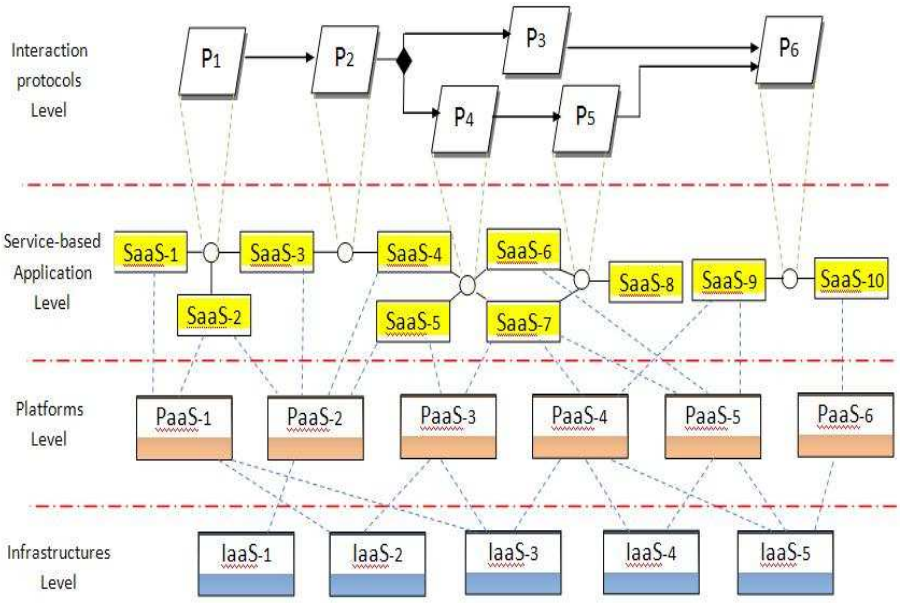

FIG. 4.1. *An Example of Business Protocols Composition*

In the example presented in figure 4.1, we have two partial Compositability relationships: (P_3, P_6) and (P_5, P_6).

Definition 5. (Full Compositability). Two protocols P_k and P_r are called Full Compositability *iff* $\forall x_i \in (P_r^{in})_\theta(C), \exists\ y_j \in (P_k^{out})_\theta(C) \mid (x_i, y_j)$, which means that the protocol P_r must be (immediately) executed after the protocol P_k because **all** its "required elements" are offered by the protocol P_k.

In our example, we have four full Compositability relationships: (P_1, P_2), (P_2, P_3), (P_2, P_4) and (P_4, P_5).

We note here that the partial (full) compositability relationship is not commutative. So, if a protocol P_k has a partial (full) compositability relationship with a protocol P_r, it does not means that P_r has a partial (full) compositability relationship with P_k .

Proposition. Let P $=\{P_1, P_2, \ldots, P_m\}$ a set of business protocols. The set P constitutes a meaningful composition if:

$$\forall P_k \in (P - P_{init}{}^3), \forall x_i \in (P_k^{in})_\theta(C), \exists\ y_j \in (P_r^{out})_\theta(C) \mid (x_i, y_j), \text{ where } r, k \in [1, m] \text{ and } r \neq k.$$

This proposition states that, if we have a set of protocols P $=\{P_1, P_2, \ldots, P_m\}$ where all their "required elements" are offered (by other protocols), this means that all the protocols of P can be executed.

In the next section we present a set of operators that allows the creation of new value-added protocols using existing ones as building blocks. Sequence, alternative, iteration, and parallel are typical constructs specified in the control flow.

4.3. Protocols Composition Operators. We describe below the syntax and semantics of the protocols composition operators. The operators were chosen to allow common and advanced protocols combinations. The set of new protocols can be defined by the following grammar in BNF like notation:

$$P ::= \quad \varepsilon \quad | \quad P \rightarrow P \quad | \quad P \diamondsuit P \quad | \quad P \blacklozenge P \quad | \quad P \curvearrowleft \quad | \quad P \xrightarrow{e} P$$

Where:

Empty Protocol. The empty protocol ε is a protocol that performs no operation. It is used for technical and theoretical reasons.

[3] $P_{init} \in$ P and represent the initial protocols, which their "required elements" are provided by external events

Sequence Operator. $P_1 \rightarrow P_2$ represents a composite protocol that performs the protocol P_1 followed by the protocol P_1, i.e., $\rightarrow$ is an operator of Sequence.

This is typically the case when a protocol depends on the output of the previous protocol. For example, the protocol *Payment* is executed after the completion of the protocol *Delivery*.

Formally, $P_1 \rightarrow P_2$ is represented by the Petri net shown in Figure 4.2-a. In this case, the two protocols are connected by a transition and n arcs, where n is a number of the participants in the protocols P_1 and P_2.

Choice Operator. $P_1 \diamondsuit P_2$ represents a composite protocol that behaves as either protocol P_1 or protocol P_2. i.e., $\diamondsuit$ is an Alternative (or a Choice) operator.

Figure 4.2-b shows a Petri net representation of the alternative operator, so that only one protocol can be executed. In this case, each basic protocol is associated to a transition.

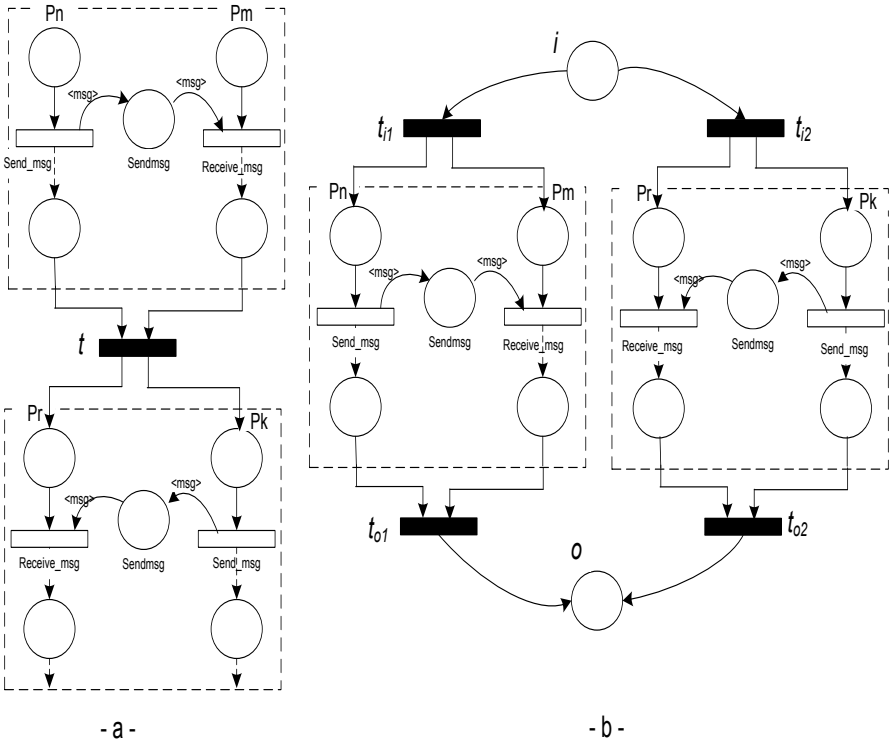

FIG. 4.2. *Protocols Operators: (a) Sequence and (b) Alternative*

Parallelism Operator. $P_1 \blacklozenge P_2$ represents a composite protocol that performs the protocol P_1 and P_2 independently from each other. i.e., $\blacklozenge$ is a Parallelism operator.

This operator which models concurrency protocols execution is represented by means of parallel case or multi-threading in Petri net (see figure 4.3-a). In fact, we add one transition and n arcs, where n is the number of the participants in the protocols P_1 and P_2.

Iteration Operator. $P\,\circlearrowleft$ represents the execution of a protocol followed a certain number of times by itself. i.e., $\circlearrowleft$ is an Iteration operator.

In Petri net (see figure 4.3-b), an iteration has a transition that is connected to all the participants places in the protocol P. So we add a transition and an arc from the end place to this transition and n arcs from this transition to the beginning participants places in the protocol P.

Refinement Operator. $P_1 \xrightarrow{e} P_2$ represents a composite protocol that behaves as P_1 except for an event e in P_1, P_2 can be executed, then P_1 can resumes its execution. i.e., $\xrightarrow{e}$ is a Refinement operator.

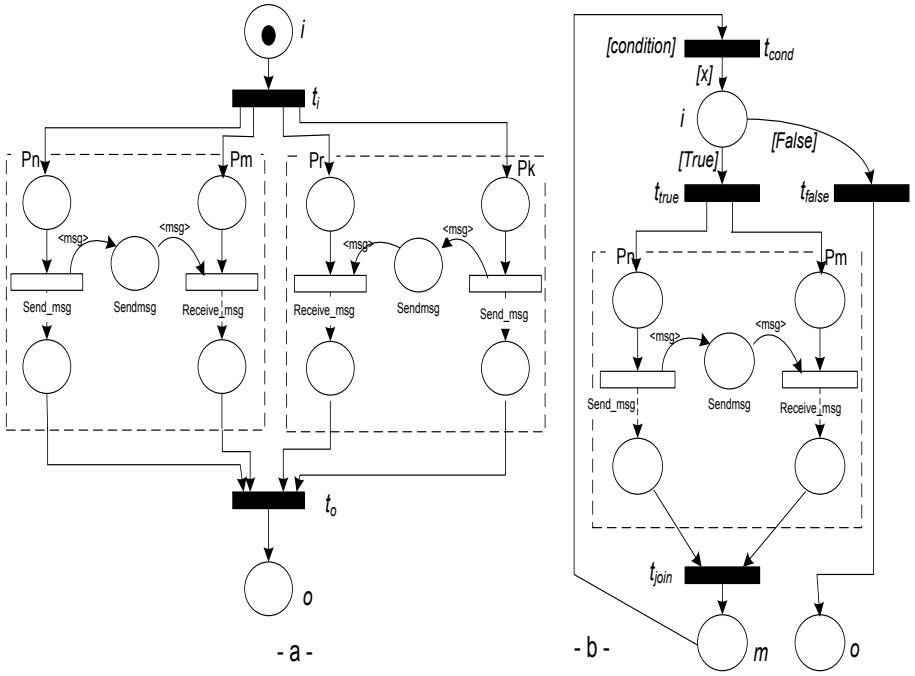

FIG. 4.3. *Protocols Operators: (a) Parallelism and (b) Iteration*

The refinement operator, in which protocols are replaced by more detailed non empty protocols. Refinement is the transformation of a design from a high level abstract form to a lower level more concrete form hence allowing hierarchical modeling.

Figure 4.4 shows an example of a refined protocol $P_1 \xrightarrow{e} P_2$ where the event $e = Receive\_msg\_1$. The refined protocol behaves as P_1 except for the event e in P_1, P_2 can be executed, then P_1 can resumes its execution.

5. Verification of Composite Protocol Correctness . A composite business protocol is a system that consists of several conceptually autonomous but cooperating units. It is difficult to specify how this system should behave and ensure that it behaves as required by the specification. The reason is that, given the same input and initial state, the system may produce several different outputs.

Business protocols interact with each other for conducting a specific task although they are created. Since IP which contain errors may cause inconsistance in the system, it is thus important to analyze the composite protocols before they are put into operation.

However, analyzing an interaction protocol is a significantly phase. Petri nets allow us to validate and evaluate the usability of a system by performing automatic and/or guided executions. Moreover, by applying other analysis techniques it is possible to verify general properties, such as absence of deadlocks and livelocks, and application-specific properties.

The transformation into Petri net is performed using our tool IP2CPN. The formal verification is performed using the CPN-AMI Petri net modeling and model checking environment [17]. From the Petri net model, we can explore its state space and look for deadlock (state from which no progression is possible), look for inconsistent state or test for more complex logical formulas.

General properties deal with application-independent concerns. These properties include boundness, liveness and deadlock-freedom. The first two properties were investigated to validate and debug the model and to provide insight into CPN behaviour.

We are interested to define the minimal conditions for the composite protocol correctness. The conditions we impose are:

1. The net is deadlock free.

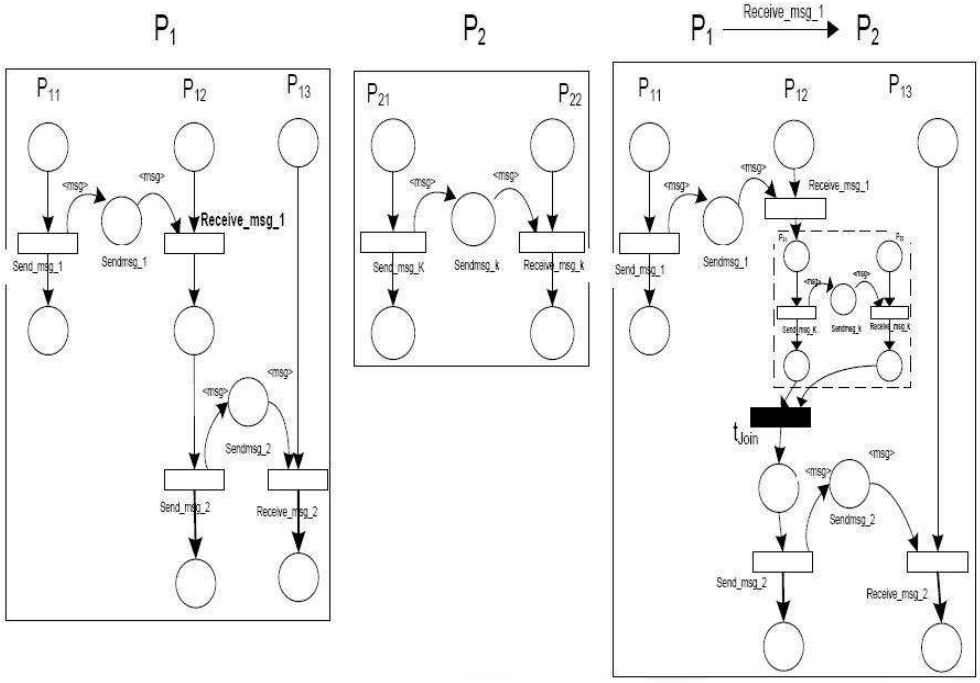

FIG. 4.4. *An Example of Protocol Refinement*

2. For any reachable marking M in the Petri net there is an execution sequence from M such that all the component protocols will still be able to terminate correctly their execution.
3. The communication places is bounded.

From the state space, we can look for deadlocks. The Petri net is deadlock-free if and only if all the role processes of the *final* protocols are terminated. More precisely, let $term_r$ be the termination place for each role process r acting during the final interaction protocols ($r \in P_{fin}$) and let $endsta$ be the terminal states of the state space.

$$\forall r \in P_{fin}, term_r \in endsta \Leftrightarrow the\,net\,is\,deadlock\,free$$

This formula states that it is eventually true that each role r will be at its termination point labeled with $term_r$. However, a role process that ends in the state r where, $r \notin \{term_r\}$ represents an active role: termination in a state where roles are active indicates a violation.

Also, we require that the participant protocols should not be allowed to send an infinite number of messages to the $Sendmsg_i$ (places that correspond to the exchanged messages in the Petri nets). So, each communication place $Sendmsg_i$ may has at most one token. This corresponds to the maximum capacity of the message buffer (ie one).

6. Related Work. Over the last few years, the prosperous research on collaborative enterprises applications has led to a multitude of results. In this section, we overview major techniques that are most closely related to our approach.

6.1. Cloud Computing Based Approaches. Today, large technology vendors as well as open-source software projects both address the hybrid Cloud market and are developing virtual infrastructure management tools to set-up and manage hybrid Clouds [21].

The Reservoir architecture [32] aims to satisfy the vision of service oriented computing by distinguishing and addressing the needs of service providers and infrastructure providers. Service providers interact with the

end-users, understand and address their needs. They do not own the computational resources; instead, they lease them from infrastructure providers which interoperate with each other creating a seamlessly infinitive pool of IT resources.

The VMware workstation [12] offers live migration of virtual appliances and machines between data centers and allows service providers to offer IaaS while maintaining compatibility with internal VMware deployments.

HP [15] provides three offerings for hybrid Cloud computing: HP Operations Orchestration for provisioning, HP Cloud Assure for cost control, and HP Communications as a Service for service providers to offer small businesses on-demand solutions. The Cloud-based HP Aggregation Platform for SaaS streamlines operations for both the service provider and the businesses customer by automating processes such as provisioning, activation, mediation charging, revenue settlement and service assurance.

Model-driven approaches are also employed for the purpose of automating the deployment of complex IaaS services on Cloud infrastructure. For instance, in [26], authors propose the mOSAIC Ontology and the MetaMORP(h)OSY methodology for enabling model driven engineering of Cloud services. The methodology uses model driven engineering and model transformation techniques to analyse services. Due to the complexity of the systems to analyse, the mOSAIC Ontology is used in order to build modelling profiles in MetaMORP(h)OSY able to address Cloud domain-related properties.

When examining the Cloud based approaches we observe a recurrent theme. They do not allow for easy extensibility or customization options. Better ways are necessary for Cloud service consumers to orchestrate a cohesive Cloud computing solution and provision Cloud stack services that range across networking, computing, storage resources, and applications from diverse Cloud providers.

The work presented in this paper is considered as a first step toward AIPaaS (Agent Interaction Protocols as a Service). The AIPaaS approach is based on the idea of reducing the complexity involved when developing a composite application. In our work, the knowledge obtained during Cloud services composition is stored, reused, and shared. In fact, we proposed a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols. Our approach provides a set of operators that allows the creation of new value-added protocols using existing ones as building blocks.

6.2. Web Services Based Approaches. With the growing trends of service oriented computing, composition of Web services has received much interest to support flexible inter-enterprise application integration. As stated in [24] and [33], current efforts in Web services composition can be generally grouped into three categories: manual, automatic, and semi-automatic composition.

By manual composition, we mean that the composite service is designed by a human designer (i.e., service provider) and the whole service composition takes place during the design time. This approach works fine as long as the service environment, business partners, and component services do not or rarely change. On the other hand, automatic service composition approaches typically exploit the Semantic Web and artificial intelligence planning techniques. By giving a set of component services and a specified requirement (e.g., user's request), a composite service specification can be generated automatically [9].

However, realizing a fully automatic service composition is still very difficult and presents several open issues [24], [9], [11]. The basic weakness of most research efforts proposed so far is that Web services do not share a full understanding of their semantics, which largely affects the automatic selection of services.

There exist some research efforts that encourage manual and automatic compositions [29] [25]. Instead of coupling component services tightly in the service model, such approaches feature a high-level abstraction (e.g., UML activity model, protocol specifications, and interface) of the process models at the design time, while the concrete composite services are either generated automatically using tools or decided dynamically at run time (e.g., BPEL4WS [22]).

Our proposition is similar to these approaches in the sense that we also adopt a semi-automatic approach for application integration. The collaboration scenario is specified as interaction protocols not high-level goals and the component applications are selected, at run time, based on the protocol specification specified at the design time.

Also, the Web services related standards for services composition and interoperability, such as the BPEL4WS are lower level abstractions than ours since they specify flows in terms of message sequences. Also, they

mix interaction activities and business logic making them unsuitable for reuse. In contrast to our approach, the BPEL4WS elements are only used to specify messages exchanges between the different business partners. Afterwards, this specification is used by agents to enact the integration of business processes at run time. Agents have the capability to dynamically form social structures through which they share commitments to the common goal. The individual agents, through their coordinated interactions achieve globally coherent behavior; they act as a collective entity known as a multiagent system. In our previous work [6] [3], we have explored the relationship between Web services, multiagent systems and enterprise application integration.

6.3. Agent Based Approaches. Multiagent systems are a very active area of research and development. In fact, several researchers are working at the intersection of agents and enterprise systems.

For example, Buhler et al. [13] summarize the relationship between agents and Web services with the aphorism Adaptive Workflow Engines = Web Services + Agents: namely, Web services provide the computational resources and agents provide the coordination framework. They propose the use of the BPEL4WS language as a specification language for expressing the initial social order of the multi-agent system. [13] does not provide any design issues to ensure the correctness of their interaction protocols.

Driven by the motivation for reuse of interaction protocols, [35] and [19] consider protocols as a modular abstractions that capture patterns of interaction among agents. In these approaches, composite protocols can be specified with a Protocol Description Language (such as: CPDL[4] or MAD-P[5]). Although formal, [35] and [19] do not not provide any step for the verification of the composite protocols.

Agent-oriented software methodologies aim to apply software engineering principles in the agent context e.g. Tropos, AMCIS, Amoeba, and Gaia. Tropos [10] [30] and AMCIS [5] [4] differ from these in that they include an early requirements stage in the process. Amoeba [18] is a methodology for business processes that is based on business protocols. Protocols capture the business meaning of interactions among autonomous parties via commitments. Gaia [14] differs from others in that it describes roles in the software system being developed and identifies processes in which they are involved as well as safety and liveness conditions for the processes. It incorporates protocols under the interaction model and can be used with commitment protocols.

Our methodology differs from these in that it is aimed at achieving protocol re-usability by separation of protocols and business rules. It advocates and enables reuse of protocols as building blocks of business processes. Protocols can be composed and refined to yield more robust protocols.

7. Conclusion and Future Work. In this paper, we presented a formal framework that allows the verification of the conformance between interaction protocols. The presented framework is based on our previous work [7] [8] that provides a formal model for the composition of local participant implementations.

The key feature of this framework is the ability to model and formally verify composition of business protocols, where particular protocols may then be selected and composed to support a new business task. The proposed framework is based on Petri net for composing interaction protocols. The formal semantics of the composition operators is expressed in terms of Petri nets by providing a direct mapping from each operator to a Petri net construction. In addition, the use of a formal model allows the verification of properties and the detection of inconsistencies both within and between protocols.

An agent-based system that supports the work is currently being developed. The proposed system has been designed to enable interoperability and cross-Cloud application management. It solves the interoperability issues between heterogeneous Cloud services environments by offering a harmonized API. Also, it enables the deployment of applications at public, private or hybrid multi-Cloud environments.

Acknowledgements. The author would like to thank the anonymous reviewers for their valuable comments and suggestions, which were helpful in improving the paper.

REFERENCES

[4]CPDL: Communication Protocol Description Language
[5]MAD-P: Modular Action Description for Protocols

[1] B. Bauer, J. P. Müller, and J. Odell, *Agents and the UML: A Unified Notation for Agents and Multi-agent Systems*, in Agent-Oriented Software Engineering II, Second International Workshop, AOSE'01, vol. 2222 of LNCS, Springer, 2001, pp. 148–150.

[2] B. Bauer and J. Odell, *UML 2.0 and agents: how to build agent-based systems with the new UML standard*, International Journal of Eng. Appl. of AI, 18 (2005), pp. 141–157.

[3] D. Benmerzoug, *Agent approach in support of enterprise application integration*, International Journal of Computer Science and Telecommunications, 4 (2013), pp. 47–53.

[4] D. Benmerzoug, M. Boufaida, and Z. Boufaida, *Developing Cooperative Information Agent-Based Systems with the AMCIS Methodology*, in IEEE International Conference on Advances in Intelligent Systems: Theories and Application, Luxembourg, November 2004, IEEE press.

[5] D. Benmerzoug, M. Boufaida, and Z. Boufaida, *From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems: An Agent-Based Approach*, in OTM Workshops, vol. 3292 of LNCS, Larnaca, Chypre, October 2004, Springer, pp. 495–506.

[6] D. Benmerzoug, M. Boufaida, and F. Kordon, *A Specification and Validation Approach for Business Process Integration based on Web Services and Agents*, in Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, In conjunction with ICEIS 2007, NSTIIC press, 2007, pp. 163–168.

[7] D. Benmerzoug, F. Kordon, and M. Boufaida, *A Petri-Net based Formalisation of Interaction Protocols applied to Business Process Integration*, in Advances in Enterprise Engineering I, 4th International Workshop on Enterprise & Organizational Modeling and Simulation (EOMAS'08), vol. 10 of LNBIP, Montpellier, France, June 2008, Springer, pp. 78–92.

[8] D. Benmerzoug, F. Kordon, and M. Boufaida, *Formalisation and Verification of Interaction Protocols for Business Process Integration: a Petri net Approach*, International Journal of Simulation and Process Modelling, 4 (2008), pp. 195–204.

[9] D. Berardi, G. D. Giacomo, M. Mecella, and D. Calvanese, *Automatic composition of process-based web services: a challenge*, in Proc. 14th Int. World Wide Web Conf. (WWW'05), 2005.

[10] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, *Tropos: An Agent-Oriented Software Development Methodology*, International journal of Autonomous Agents and Multi-Agent Systems, 8 (2004), pp. 203–236.

[11] J. Bronsted, K. M. Hansen, and M. Ingstrup, *Service composition issues in pervasive computing*, IEEE Pervasive Computing, 9 (2010), pp. 62–70.

[12] E. Bugnion, S. Devine, M. Rosenblum, J. Sugerman, and E. Y. Wang, *Bringing virtualization to the x86 architecture with the original vmware workstation*, ACM Trans. Comput. Syst., 30 (2012), pp. 12:1–12:51.

[13] P. A. Buhler and J. M. Vidal, *Towards adaptive workflow enactment using multiagent systems*, International Journal On Information Technology and Management, 6 (2005), pp. 61–87.

[14] L. Cernuzzi, A. Molesini, A. Omicini, and F. Zambonelli, *Adaptable multi-agent systems: the case of the gaia methodology*, International Journal of Software Engineering and Knowledge Engineering, 21 (2011), pp. 491–521.

[15] D. Collins, *Communications as a service for midsize businesses*, 2009.

[16] A. Copie, T.-F. Fortis, V. I. Munteanu, and V. Negru, *Datastores supporting services lifecycle in the framework of cloud governance*, Scalable Computing: Practice and Experience, 13 (2012), pp. 251-7.

[17] CPN-AMI:, *http://move.lip6.fr/software/cpnami/*.

[18] N. Desai, A. K. Chopra, and M. P. Singh, *Amoeba: A Methodology for Fodeling and Evolving Cross-Organizational Business Processes*, Journal of ACM Trans. Softw. Eng. Methodol., 19 (2009).

[19] N. Desai and M. P. Singh, *A modular action description language for protocol composition*, in Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, AAAI Press, 2007, pp. 962–967.

[20] C. Girault and R. Valk, *Petri Nets for Systems Engineering, A Guide to Modeling, Verification, and Applications*, Springer Verlag, July 2002.

[21] S. V. Hoecke, T. Waterbley, J. Devos, T. Deneut, and J. D. Gelas, *Efficient management of hybrid Clouds*, in The Second International Conference on Cloud Computing, GRIDs, and Virtualization, Rome, Italy, September 2011, pp. 167–172.

[22] SAP, Siebel Systems, IBM, Microsoft, *Business process execution language for web services version 1.1*, tech. rep., 2003.

[23] T. F. R. market:, *http://www.forrester.com/*.

[24] N. Milanovic and M. Malek, *Current solutions for web service composition*, IEEE Internet Computing, 8 (2004), pp. 51–59.

[25] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari, *Declarative specification and verification of service choreographiess*, International journal of ACM Transactions on the Web, 4 (2010).

[26] F. Moscato, B. D. Martino, and R. Aversa, *Enabling Model Driven Engineering of Cloud Services by using mOSAIC Ontology*, Scalable Computing: Practice and Experience, 13 (2012). pp. 29–44.

[27] D. K. Nguyen, F. Lelli, M. P. Papazoglou, and W.-J. van den Heuvel, *Blueprinting Approach in Support of Cloud Computing*, International Journal of Future Internet, 4 (2012), pp. 322–346.

[28] OCL:, *Object constraint language.* (www.omg.org/cgi-bin).

[29] M. P. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, eds., *Service Research Challenges and Solutions for the Future Internet - S-Cube - Towards Engineering, Managing and Adapting Service-Based Systems*, vol. 6500 of Lecture Notes in Computer Science, Springer, 2010.

[30] L. Penserini, T. Kuflik, P. Busetta, and P. Bresciani, *Agent-based organizational structures for ambient intelligence scenarios*, Ambient Intelligence and Smart Environments, 2 (2010), pp. 409–433.
[31] P. Mell, and T. Grance, *The NIST Definition of Cloud Computing*, 2009.
[32] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Cáceres, M. Ben-Yehuda, W. Emmerich, and F. Galán, *The reservoir model and architecture for open federated cloud computing*, IBM Journal of Research and Development, 53 (2009), pp. 535–545.
[33] Q. Z. Sheng, B. Benatallah, Z. Maamar, and A. H. H. Ngu, *Configurable composition and adaptive provisioning of web services*, IEEE T. Services Computing, 2 (2009), pp. 34–49.
[34] S. Subashini and V. Kavitha, *A survey on security issues in service delivery models of cloud computing*, J. Network and Computer Applications, 34 (2011), pp. 1–11.
[35] B. Vitteau and M.-P. Huget, *Modularity in interaction protocols*, in Advances in Agent Communication, vol. 2922 of LNCS, Springer, 2004, pp. 291-9.
[36] FIPA - Foundation for Intelligent Physical Agents:, *FIPA Communicative Act Library Specification.* (http://www.fipa.org/specs/XC00037/).
[37] J. Abreu, L. Bocchi, J.L. Fiadeiro, and A. Lopes, *Specifying and Composing Interaction Protocols for Service-Oriented System Modelling*, in International Conference on Formal Techniques for Networked and Distributed Systems - FORTE'07, vol. 4574 of LNCS, Springer, 2007, pp. 358-373.

Edited by: Viorel Negru and Daniela Zaharie
Received: May 25, 2013
Accepted: Jul 9, 2013

DOI 10.12694/scpe.v14i2.844
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. 111–129. http://www.scpe.org

ESTABLISHING SEMANTIC CLOSENESS IN AN AGENT-BASED TRAVEL SUPPORT SYSTEM

MARIUSZ MESJASZ*, MARCIN PAPRZYCKI†, AND MARIA GANZHA‡

Abstract.
Agent system that supports needs of travelers is a classic example of an application of personal agents. This paper re-evaluates assumptions behind the design and implementation of an agent-semantic travel support system (TSS) that was proposed in 2006. The goal is to modify these assumptions to match them with the current user-demands and adjust to the technological changes. On the basis of this analysis, the design of a new TSS is introduced. The key element of the proposed system is semantic matchmaking, which is used to establish, which travel services are to be suggested to the user (an in which order). Therefore, we describe in detail the proposed matchmaking algorithm and its implementation, and illustrate its work through selected examples.

Key words: Software agents, JADE, ontological matchmaking, recommender system, resource management, resource matching, semantic closeness

1. Introduction. One of the areas where modern technology is making its mark is the "world of travel." Nowadays, tourist can use his smart-phone to locate restaurants in the nearby area. Then, based on their ranking, published on a website (available through a smart-phone application), she can choose the one that matches her food preferences, while being highly recommended by others. Similarly, she can use web-based services to find means of transportation to reach the selected restaurant.

Unfortunately, this scenario sounds easier / nicer than it is in reality. First, as a matter of fact, tourist may be "forced" to visit several websites to gather the necessary information. Second, she has to spend time comparing the results to make an educated choice. To avoid the hassle, she may make the *simplest* selection (possibly, "contradictory" to his preferences), instead of looking for the *best* one. For example, a hungry tourist may choose an easily-accessible fast food place (even though she does not like fast-food), over restaurants that serve her favorite cuisine. The problem remains even when a tourist is in a hotel with an Internet access and spare time to check "everything." Here, she will spend time searching and comparing restaurants, rather than doing "something else." This illustrates the need for a travel support system that will be able to "intelligently" work for the user. For example, a tourist should be able to make a selection from a short list of candidates, based on cross-referenced sources, and containing only results that match her preferences.

2. State of the art in related areas. Let us start by considering state of the art in three areas related to travel support. First, we will consider general web-based travel services. Second, we will discuss travel services related to mobile technologies. Finally, we will comment on the state of ontology-based personal assistants.

2.1. Examples of travel-related systems. There are several services and applications, which facilitate travel-related information. Here, we discuss some of them, including two no longer developed systems - the "original" Travel Support System (predecessor of the one described in this paper), and the Chefmoz (ontology-based collection of restaurants). Despite the fact that they are no longer active, they provide a useful source of information and reflection.

Travel Support System (TSS; 2000–2008) The preliminary idea of the Travel Support System was introduced in the MS Thesis from 2000 [29]. The TSS was an agent-based system responsible for collecting information available on the Internet, storing it in an RDF database, and creating personalized answers to users' queries. The initial idea was explored in a series of publications that appeared between 2001 and 2008 (available at [18]). The working system prototype was released in 2006, and is still available at the SourceForge repository [19].

Interestingly, the need to redesign the system became evident immediately after its initial implementation. Most important lessons learned in the process of its development, implementation and evaluation

*Warsaw University of Technology, Department of Mathematics and Information Sciences
†System Research Institute Polish Academy of Sciences and Warsaw Management Academy (marcin.paprzycki@ibspan.waw.pl)
‡System Research Institute Polish Academy of Sciences and University of Gdask

were summarized in [28]. The publication pointed out to several design flaws concerning utilization of agents in the TSS system.

However, none of these problems have been resolved, as the development of the TSS stopped. Furthermore, due to broken software dependencies, the system is no longer operational. Therefore, the new implementation of the system proposed in this work started from requirements analysis, taking into account the previously acquired knowledge and the development of new technologies.

Chefmoz (2000 2011) Chefmoz was a branch of the Open Dictionary Project [9] and contained an online directory of restaurants (and their reviews). The collection covered more than 300,000 restaurants from 142 countries. In 2009, ChefMoz had become the largest (available on the Internet) global directory of restaurants. Similarly to the Open Dictionary Project, the Chefmoz provided its data as a hierarchical ontology scheme (represented as "low quality" RDF triples; where, low quality means that they did not actually adhere to ontological standards, contained large number of non-RDF tags, and could not have been imported into ontology development platforms, e.g. Protege, without considerable amount of extra manual work). Due to the persistent technical difficulties and lack of funding, the Chefmoz project was announced "dead" in 2011.

Booking.com Booking.com [3] is a hotel reservation portal. It allows its users to browse hotels, book rooms, and post their opinions. Website has a number of important features such as a large database of hotels (over 250,000 entries), opinions posted by real hotel guests, and worldwide accessibility (consumer support in more than 40 languages). Here, Booking.com represents a broad class of hotel-focused services (information aggregators) such as: hotels.com, venere.com, etc.

Despite the fact that Booking.com is a high quality service, we have to keep in mind that it is limited to hotel information. Therefore, it cannot answer complex queries (e.g. find hotels located near museums or art galleries). Moreover, Booking.com does not provide personalized recommendations. For example, it can be conjectured that a many users could find value in offers adjusted to their preferences, instead of a regular newsletter containing "statistically good deals" (for example, the cheapest prices).

Social Networks Many social networks could be a source of travel-related information and affect decisions where to spend time. The two most popular social networks of today are Facebook [5] and Twitter [21]. Users of both, can post comments, upload photos and share information that can influence travel decisions. For example, if our friends frequently visit a place, publish cool pictures, and localize the place on the map, it is likely that we will consider going there in the future. Observe that the TripAdvisor [20], a web page similar to Booking.com, uses this idea and integrates its services with Facebook. Customers may connect to their Facebook accounts and see offers from places visited by their friends.

Although this is an interesting use of social networks, this approach has important drawbacks (from the point of view of the development of the TSS). First, these services allow users to share all kinds of information (not only related to travel). For example, the user can post on Facebook not only information about a restaurant which, in her opinion, serves the best pepperoni pizza (travel information about a certain place), but also a comment about the current political situation in Andorra (information barely related to the TSS). Secondly, in the social media, the term "recommendation" is not appropriate (for example, while messages may encourage a user to go to a specific location or participate in an event, none of them was an actual recommendation). Finally, systems based on current social networks, would have to be substantially augmented to take into account the user profile. In their natural form they would be most likely based on opinions from "friends" (leading to questions as to the actual nature of social network acquaintances). Therefore, it might not be easy to provide recommendations that would match user preferences.

Yelp! Yelp! [25] is a web catalog of local businesses. It operates as a social network service – its content is filled by advertisers (local businesses) and ordinary users (reviewers and potential customers). The drawback is that the system does not provide personalized, or context-driven, recommendations. Therefore, users receive a newsletter, which is loosely connected to their profile (for example, it is based on the city where they live). As the result, the user receives a limited number of local offers that may or may not match his preferences. Yelp! is also a medium for friend-based recommendations. As a result, offers depend on the "external" recommendations, not on users' preferences. Finally, user recommendations

have to be completed manually, which takes time that the user may not be willing to spend.

Google Now Google Now [6] is an intelligent personal assistant provided by Google. The most important feature of Google Now is the ability to display information obtained in the interaction with the Google services, e.g. weather, traffic information, or dishes recommended in a restaurant. Information provided by the application is selected on the basis of Google collected data, such as Web History and location services (for instance, the GPS). Although Google Now seems to be similar to the Travel Support System, there are some differences. First of all, the TSS is focused on supporting needs of "travelers." Google Now, on the other hand, promotes use of Google services. Second, Google Now forces users to store personal information on Google's servers. Despite the fact that this is a useful feature (for example, if a mobile device is stolen), users have no other choice. Finally, as seen above, there is more to the world of travel than the Google services, and user should not be forced to use only one set of services, regardless how good it is.

2.2. Travel-related mobile applications. Let us now describe few smartphone-based applications that have been developed to support needs of travelers. First of all, it is worth to notice that most webased services, mentioned in the previous section, provide their own mobile applications. Unfortunately, such mobile applications often do not have additional features beyond the ability to quickly browse content of the web page of a given service on the mobile device. Thus, they suffer the same limitations as tehir web-based versions. For example, the mobile application [4] provided by Booking.com helps to quickly navigate through the list of available hotels. However, no one can use this application to cross-reference hotels with a list of museums. Therefore, similarly to Booking.com, the application provides only data related to hotels.

Secondly, due to the openness of application stores, for popular mobile operating systems like Google Android (Play Store) and iOS (iTunes App Store), a large number of travel-related applications has been developed. For example, OpenRice Hong Kong [11] (version 2.7.0) developed by Openrice Group Inc. [10] is a dining guide, which provides information about more than 40,000 OpenRice restaurants across Hong Kong. Unfortunately, according to its user reviews, it has some major bugs. Users have pointed out frequently crashes, poor quality on HD displays and persistent error messages. Yet another example, iPic Theaters [8] (version 1.2), developed by the iPic Entertainment [7], is a mobile application which helps its users buy tickets. Similarly to the OpenRice, its users complain about poor application performance. Specifically, the application can crash after the credit card information is provided. As a result, its users cannot be sure if the reservation has been processed. These two examples illustrate how hard it is to find a reliable travel-related applications.

Finally, with the development of the Internet, the idea of Semantic Web [16] is becoming ever more important. Here, the final outcome should be meta-level interoperability and understanding of all Internet-stored data. However, there are no well-known mobile applications that support this idea. All applications operate on their own data (for example, Booking.com, Yelp!) or connect to free data sources (sometimes characterized by low-quality content). Therefore, there is no unified way to obtain complex information (for example, cross-referencing monuments with restaurants). The user is still forced to use different travel-related applications in order to find some information separately.

Summarizing, there exists vast number of travel-related mobile applications, but very often they suffer from important limitations. These limitations either come from the source web-based services (the application is simply a front-end) or are caused by utilization of inaccurate data sources. Moreover, many developers failed to deliver a user-friendly, high-quality software. Overall, it can be stated that at the time of writing of this paper, application stores are full of low-quality mobile applications. Finally, despite the fact that Semantic Web could provide many benefits (unified access to the information, and cross-referencing), this idea is not utilized.

2.3. Semantic personal assistants. Let us now look into ontology-driven travel assistants. Ideas in this direction have been formulated already in 2003 (or, possibly, earlier). For instance, in [1] authors discuss the general idea of a semantic personal agent. Here, one of the use case application areas is travel support. Paper refers multiple technologies of the time and mentions DAML-based ontologies as the way of representing knowledge. However, it seems that the project did not progress beyond a very limited prototype.

In 2008, authors of [17] discuss a Smart Travel Service Adviser (STSA). The STSA is an agent-based solution for searching travel services, and it is focused on the User Satisfaction Level (USL). Here, RDF triples were to be used to represent knowledge. They were to be combined with Jena middleware for semantic data

storage and management, and with SPARQL for semantic queries. In this way the STSA project resembled the original TSS. Here, again, it does not seem that the project progressed beyond the initial prototype.

The lack of *actual* progress in the area of semantic personal agents for travel support is illustrated by the short article from May, 2011 [2]. Here, Larry Smith, repeats well known mantra that the semantic web will produce great results. However, he is not pointing to any existing / being developed applications.

One could assume that the problem is with development of semantic agents to be used for travel support. However, this is not the case. Let us mention just two examples. The EU funded project "A Platform for Organisationally Mobile Public Employees" (Pellucid; [12, 13, 14]) aimed at development of semantic personal agents to support worker mobility. Currently, all that remains are the web pages (some of them already returning the 404 error) and published articles. No actual application of Pellucid Semantic Assistants can be located.

Second, let us mention a Polish national Project WKUP (Virtual consultant for public services; [22, 23]). The aim of the project was to create an agent-semantic assistant (avatar) to support users of public services. Apparently, in 2009, in the Radlin county there was a trial run of the system. However, currently the www site of the trial is a broken link, and no other application of software developed within the WKUP project can be found.

In summary, the idea of semantic (travel oriented) personal assistants is very popular since at least early 2000. However, regardless of the public funding devoted to research, and large number of publications devoted to the subject, it is practically impossible to locate working prototypes (not to mention real-world applications).

3. Proposed System – Preliminary Considerations. Let us recall that systems outlined above have certain shortcomings. The information that they provide is either very narrow (e.g. Booking.com), or very robust but obscure (e.g. social networks and Google Now). Based on analysis of their functions, we can formulate the main features of our Travel Support System. However, before presenting the list of requested features, let us present two use case scenarios.

Scenario 1 Jack is on a business trip in a city that he has not visited before. Spending his free time between business meetings, he decided to visit the city center. After some time of sightseeing, he became hungry. Thankfully, he has installed (on his tablet with build-in 3G modem) the Travel Support System (TSS). The system is aware that Jack is a "committed vegetarian" (this information is stored in his TSS *user profile*). Therefore, the TSS rejects a very popular restaurant serving steaks (that happens to be near-by) and recommends a little known Thai/Indian place that serves vegetarian/vegan meals.

Scenario 2 Jack recently met Jill, and began to SMS-chat with her. After Jack sends a message to Jill, asking for a date, (assuming that she agrees) his instance of the Travel Support System starts negotiations with the one running on Jill's smart phone. Since Jack is already in her contacts, her Travel Support System shares selected information about her preferences (stored in her TSS profile). As a result, Jack's TSS sends to Jill's TSS a suggestion to meet in an Indian restaurant. In response, Jack is informed that Jill will likely be satisfied with this choice since she likes Indian food (information confirmed by her TSS). Note that we do recognize privacy issues present in this scenario, but we believe that they can be appropriately addressed.

From the use case scenarios, one can derive some observations. First, the system should produce personalized recommendations, based on user preferences (stored in a TSS user profile). For example, the restaurant proposed to Jack serves the appropriate type of food – vegetarian meals. Similarly, in the second scenario, the Indian restaurant recommended to Jack and Jill serves food that they both like. Second, the system should not dependent on location (restaurant can be anywhere) or the timing (now, or in a few days). In addition, there should be no difference between a spontaneous decision (find a restaurant now) and a deliberate action (planning dinner ahead of time). Depending on the situation, the system should either produce immediate results, or take time to prepare an in-depth list of potentially interesting places.

These observations allow us to present our vision of the TSS; depicted as a use case diagram, in figure 3.1. There we represent three entities located outside of the system: the *GPS*, the *User* and the *Data Source*. They interact with the system by asking queries (*User*), providing information (*GPS*, *Data Source*) and negotiating. The main actor of the system is the *User*. The TSS is involved in five functions: *Interacting* with the *User*, *Context monitoring*, *Learning*, *Negotiating* and *Cross Referencing*.

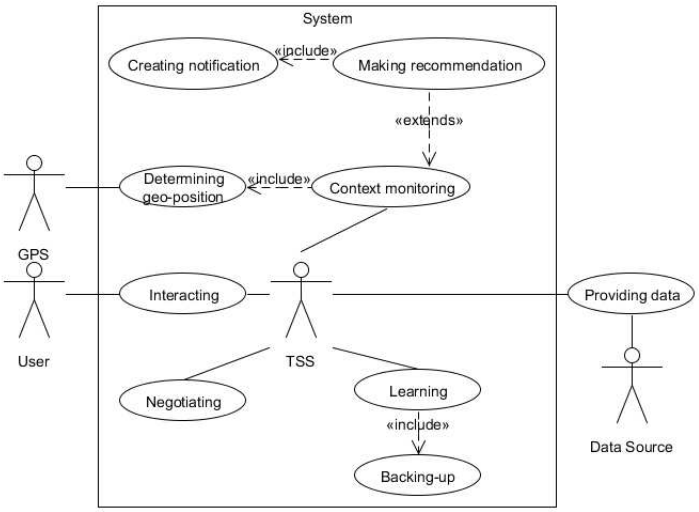

FIG. 3.1. *Use case of the Travel Support System*

The *Interacting* function involves preparing a list of results that match user preferences or correspond to the user input. It is also connected to the *Cross-referencing* and the *Learning* functions. The *Cross-referencing* function is used to obtain the results from one or more *Data Source*s. By the nature of the system, each action taken by the *User* triggers the *Learning* function. During the learning phase, the system adjusts user preferences stored in the hierarchical structure called the *user profile*. Due to this, the system is able to obtain useful information from the *Data Sources* (the information that match the user preferences). The *Context monitoring* function involves dealing with all contextual information available to the system. The most important contextual information is the geopositioning, provided by the *GPS*. However, the *Context monitoring* may also involve other types of information (for example, text messages, calendar entries, and the remaining battery power of a mobile device). The *Context monitoring* can trigger the *Making recommendation* function. The *Making recommendation* function involves proactive searching for information, based on the current user context. It is also connected to the *Cross-referencing* and the *Creating Notification* functions. The *Creating notification* function involves displaying a notification about a new set of recommendations made by the system. Here, it is assumed that this function will adjust the display format to the actual user device (used to interact with the TSS). The *Negotiating* function that connects two (or more) instances of the TSS, involves exchanging messages (in order to achieve an agreement). In the TSS, the agreement may, for instance, concern selection of a meeting place that will be satisfactory to two or more users. The *Negotiating* function is also connected to the *Checking permissions* function. The *Checking permissions* function ensures the confidentiality of shared information (for example, the local TSS will not start negotiations with a remote TSS if its owner is not in the contact list). Finally, the *Learning* function is connected to the *Backing-up* function that stores the current user profile on a remote server. This is crucial to prevent data loss when something happens to the mobile device (e.g. it is lost or stolen).

On the basis of what has been said thus far, we propose the following requirements for the TSS:

Use of mobile technology With the increasing popularity of mobile devices, and an almost ubiquitous access to high speed Internet connection, the Travel Support System should be able to take advantage of these capabilities. Thus, it should be able to run on smart phones, tablets, notebooks, as well as on desktop PCs. Furthermore, it should use services available in such devices (e.g. geo-positioning in smart phones and tablets). Finally, the system should be robust enough to allow users to select the right device for given situation (in the hotel room, the most preferable device could be a laptop, while on a street it may be a tablet or a smart-phone). However, the system should take limitations of each device into account and provide a suitable display mode for each of them.

Context-awareness In order to provide the best possible response to the user queries (and to be proactive), the system has to monitor user's contextual information. In the travel-related system, the most useful information is the geo-position (the GPS feed), which can be used most of the time (except, when the user decides otherwise). In the second scenario, an additional data source was considered: SMS messages. These messages may be used by the system to be proactive. They do not directly alter user queries, but may trigger autonomous actions within the application. For instance, a content of text messages is not included in the user data (the system is not likely to understand what it means), but it could try to involve them to produce a useful recommendation (it can query certain keywords – for example, "restaurant," "French," "dinner" – and check the results against the user profile).

Personalized Searching The result of a search has to be personalized to individual user's requirements. The requirements may be specified directly (user explicitly adds them to the query), or indirectly (the system is aware of user's previous choices and tendencies – instantiated in the profile). The system has to filter-out data contrary to the direct, or indirect, requirements and favor those which match user preferences, and are more likely to fulfill her needs. Therefore, the personalized results should be presented as a response to a query, while other results may be added to introduce variation and/or fill the space (e.g. in the case when only few services matching the query can be found).

Here, let us address the fact that the system should take into account potential changes of user preferences (and desires for something new). Thus, from time to time, it should return a result which is neutral or even contradictory to the stored preferences. The average time between such proposals may be measured by analyzing changes in user queries and responses to unusual suggestions. The contradictory result should be somehow "separated" from others (and marked appropriately). However, the system should be aware that some preferences may be very stable or practically unchangeable. For example, in the first scenario, Jack is a *committed* vegetarian, hence the system should not recommend him a steak.

Collecting important user information In order to allow personalized searching and to manage the user profile, the system must be able to collect information about the user. Observe that the user profile should be dynamically adjusted over time, based on the user interaction with the system (e.g. queries and responses, and decisions based on them).

Negotiations Objectives of negotiations may differ, depending on the functionality of the system. For example, in the original Travel Support System ([29]) negotiations between multiple instances of the system - representing travel service sellers and tourists – were proposed. There, special offers and promotions, based on user preferences, were to be negotiated. For instance, a restaurant's recommendation might also include information about availability of a free appetizer.

The new version of the system focuses on negotiations with a different objective. For instance, in the second scenario, Jack and Jill want to go out for a dinner, but none of them knows the best *common* place to go to. Therefore, the two instances of the TSS can negotiate possible place to meet.

Cross-referencing Many information services provide data pertinent only to a single topic (e.g. Booking.com provides information about hotels) and their users have to cross-reference multiple sources of information. Therefore, the system should be able to combine different sources of information, and respond to more sophisticated queries.

Online data storage and accessibility of the user profile from multiple devices Due to the fact that the TSS is targeted mainly for mobile devices (see, scenario 1 and 2), the following issues have to be taken into consideration: (a) serious risk of data loss caused by accidents while traveling (for example, losing or damaging the device), and (b) need of copying the user profile from one device to another. In the first case, we should recognize time that the system spends learning user preferences (creating a user profile). Total loss of such profile would mean that the system would have to start from scratch, which is highly undesirable. The second problem can be easily solved by connecting the device with an old user profile to one with the newest version and simply copying the profile. However, if the user forgets to copy his profile from one device to another, the system may not find the best results, with respect to her current preferences. The solution to these two problems is in online storage and synchronization. The user should be able to enable backing-up of her current profile to a remote server (e.g. to a cloud)

and uploading it whenever the current device has an obsolete profile.

Obviously, due to the limited space, we cannot address all issues described above. Therefore we have decided to focus on crucial two. First, we will briefly discuss the general agent structure of the new TSS. Second, we will describe in detail how we represent data in the system and how this data is used to make recommendations.

4. Software agents and the Travel Support System. As stated above, the new TSS expands upon the ideas presented in [19]. One of the key features introduced in the original Travel Support System was a comprehensive application of software agents [24]. However, when the resulting system was thoroughly analyzed (in [28]), several design flaws concerning usage of agents were elaborated. Let us summarize them:

- Software agents should not be used as a middleware solution, unless it is beneficial.
- Software agents should not replace currently existing technologies, unless there is a specific need for this.
- Designing an agent-based system requires finding balance between the autonomy of an agent and its need to communicate.

Unfortunately, these problems were not fixed, while the original TSS became obsolete due to software evolution (including collapse of some projects, like the Racoon), leading to unresolvable software dependencies. Therefore, a fresh start was needed for use of agents in the TSS.

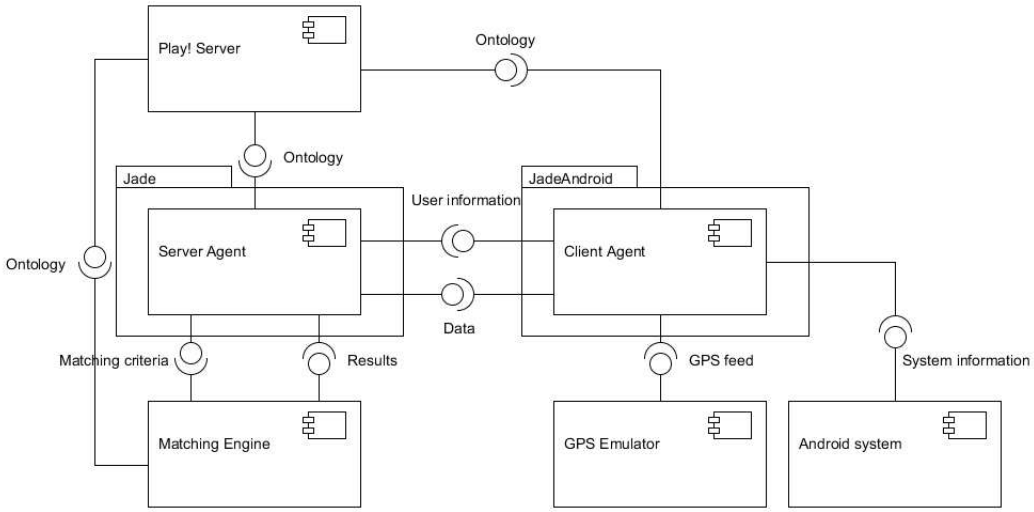

FIG. 4.1. *The component diagram of the new Travel Support System*

Figure 4.1 presents the component diagram of the proposed Travel Support System. Since one of our main assumptions was that the system will involve mobile technology, we have designed the TSS accordingly. As we can see, it is composed of two types of software agents: 1) *Client Agents*, and 2) *Server Agents*. These two types of agents perform different (and disjoint) tasks, while cooperating in order to support users.

The Client Agent runs on the mobile device (in our case on an Android-based system). First of all, the Client Agent manages the user profile on the (mobile) device. During the start-up of the system, the user profile is loaded from the server side of the system into the memory of the mobile device. Here, the agent receives an I/O interface, through which it can read and appropriately modify local user preferences.

The Client Agent is also responsible for backing-up the user profile and synchronizing it with the on-line server (the one, which runs the Server Agent). Obviously, the Client Agent monitors the user context. Since it is located on the mobile device it has access to the user data stored there. Here, let us stress, that the user is informed about this fact during the installation of the application; similarly to other applications running on the Android-based systems. The private user data includes text messages, entries in the calendar, contacts, and so on. Furthermore, the Client Agent has access to the GPS data, and the GIS software that provide the geo-spatial context.

To provide user with recommendations, the Client Agent communicates with the Server Agent. Here, we have decided that, even though the mobile devices have ever increasing power, their battery longevity and the complexity of the recommendation algorithm require that the actual data searching, cross-referencing and filtering should be implemented as a server-based solution. During communication, the Client Agent provides the Server Agent with information required to construct the search criteria, e.g. fragment of the user profile corresponding to the search query and the user contextual information (GPS coordinates and the current time). The user profile fragment contains all information related to a specific topic (for example, restaurants).

The actual search criteria are constructed by the Server Agent. Then, the Server Agent uses them in the local instance of the recommendation algorithm. The algorithm processes the search criteria and matches them against the available data sources. As a result, a set of most suitable recommendations is sent to the Server Agent. Then, the Server Agent communicates with the Client Agent who will presents the recommendation to the user.

5. Data representation in the system. Knowing the use case scenarios, the general idea behind the system, and the role of software agents, we have to consider how the information about travel services and user profiles is going to be represented. In the original TSS ([19]), ontologies represented in the RDF format [15] were used. Therefore, we have decided, for the time being, to re-use the already existing ontologies; in particular, the restaurant ontology. The assumption was that, as the system develops, a more comprehensive ontology of the world of travel will be incorporated (and/or developed if needed).

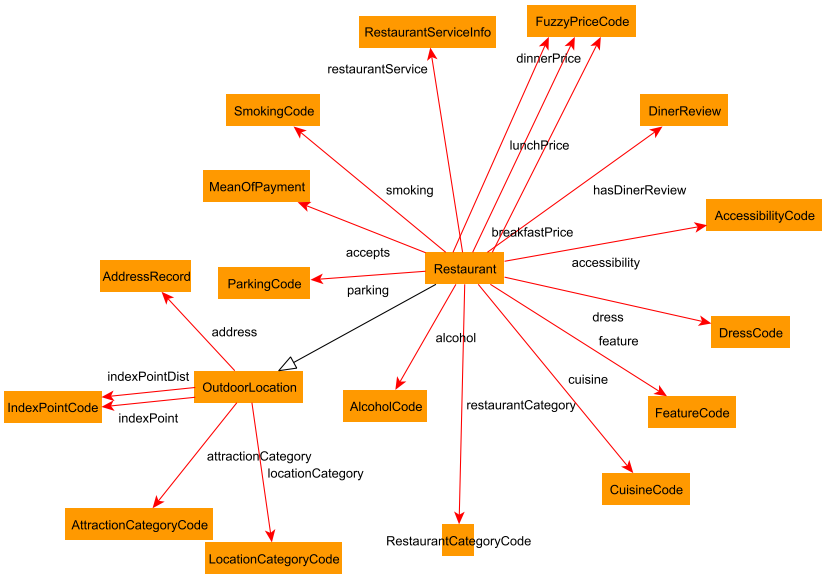

FIG. 5.1. *The structure of the Restaurant ontology*

The *Restaurant Ontology*, presented in Figure 5.1, contains the *Restaurant* class, which captures relations between the concept of the restaurant and its features. Restaurants and restaurant features are represented in the ontology as individuals (see, figure 5.1 for more details).

However, when deciding how to store the user profile, we have decided to use a simplified representation. Instead of adding additional elements to the ontology, we store pairs $(keyword, weights)$. Here, the keyword is an information object related to the *Restaurant* class, while the weight represents user preferences. Proceeding in accordance with the original TSS, we have decided to represent user preferences as numbers from the interval $[0, 1]$. This approach is based on the observation that, not only the Restaurant Ontology has few concepts, but also that not all of them have to have actual weight associated with them. For instance, if the system does not have any notion about users' opinion about Chinese cuisine, no user-specific weight can be associated with it

(and a neutral value of 0.5 has to be used). As a result, keywords for which no opinion is present do not need to be stored in the user profile.

Observe that this representation simplifies somewhat the process of preparing the recommendation, as it is relatively easy to access weights (for example, by using a hash mapping) and modify them if some adjustments are needed. Taking this into account, since Jack (from use case scenario 1) likes vegetarian food, his user profile could consist of the following pairs:

1. (*cur:Vegetarian*, 0.91)
2. (*cur:Indian*, 0.65)
3. (*cur:SteakHouses*, 0.04)
4. (*asc:SmokingArea*, 0.3)
5. (*drs:Casual*, 0.66)
6. (*rcc:FastFoodRestaurant*, 0.1)
7. (*rcc:CasualRestaurant*, 0.7)

Note that the profile reflects Jack's preferences by assigning appropriate weights to the "Vegetarian" (0.91; the largest weight) and the "SteakHouse" (0.04) cuisines. Thus the TSS system, working on Jack's mobile device will strongly favor vegetarian food over the other types of food. In addition, the steak house has the lowest priority in comparison with other cuisines (including those not listed in the Jack profile that will have the default weight – 0.5).

Finding the most suitable recommendation for Jack is going to be preformed by matching his user profile with profiles of potential restaurants. This operation is very similar to the item-to-user recommendation performed by many modern recommender systems [32]. However, this task is more complex, due to the data representation utilized in the TSS. Specifically, ontologies cover more in-depth relations between objects than the popular recommendation algorithms usually exploit. Hence, in this paper we describe, in-depth, the approach selected to make the ontology-based recommendations.

6. The semantic matchmaking algorithm. As discussed above, the central functionality of the Travel Support System is its ability to make recommendations. Since, in the TSS, all travel information is ontologically represented, recommendations can be seen as the realization of the problem of establishing semantic closeness between entities. Specifically, assuming that users' restaurant preferences are represented in his user profile, and all restaurants are instances of an ontology, the question that has to be answered is: which of the available restaurants matches the user profile the closest. Of course, since the TSS is also going to use geopositioning, an important filter may be the location. Thus the system should also be able to answer the question: which of the restaurants at a distance not larger than 1 km is matching a given user profile the closest.

To be able to respond to such queries (to establish "semantic" distance between entities), we have selected the Rhee-Ganzha algorithm [31]. This algorithm was originally proposed in 2009, but it has never been fully tested for a realistic ontology. Therefore, its implementation and application in the TSS will also allow us to establish its usability.

6.1. General description of the Rhee-Ganzha algorithm. Let us start from the general description of the algorithm. It is based on two observations. First, ontologies can be seen as a directed graph. Second, in the knowledge space, information resources do not exist in isolation from each other, but are connected through multiple relations. Such relations can be explicit or inferred. Thus, the existence, the strength, and the number of relations between two resources determine a degree of their semantic similarity (closeness). This, in turn, can be translated into a problem of establishing existence, and finding number and strength, of connections in a directed graph (on the way from the source – user, to the sink – travel entity). However, as it will be seen, this is not a simple network-flow problem, where the total throughput between the source and the sink is sought.

Obviously, the semantic closeness between the user (profile) and travel services can be used to deliver recommendations to the user. Namely, if the value of the semantic closeness between the user and the restaurant A is 0.75, while for the restaurant B it is 0.3, then the restaurant A is much more likely to match her desires. Moreover, semantic closeness can be also seen as a "degree of relatedness" between two information objects, or "how relevant one object is to another." Specifically, if two individuals are in many explicit or inferred relations, we can say that they are relevant to each other. Thus the proposed algorithm can help determine the level of semantic closeness (or relevance) to entities, which are quite different. For example, if the user likes restaurants

with smoking areas and free WiFi, then the system could use this information to: (a) create an alternative recommendation, which matches only some user preferences, or (b) recommend hotel on the basis of its having features (smoking area and free WiFI) that are liked by the user in restaurants. Taking this into account, let us summarize main assumptions behind the algorithm for calculating semantic closeness.

6.2. Core assumptions of the semantic matchmaking algorithm. The relevance (semantic closeness) calculation is based on the following three assumptions:

1. Having more relations from one object to another means that they are closer (more relevant to each other).
2. Each relation may have different importance depending on the type of the relation.
3. Even if the relations are of the same type, the weight of the connection can vary between individual objects (instances).

Since these assumptions are often referenced in the remaining parts of the paper, let us elaborate them. The first assumption is very intuitive. For example, let us consider three restaurants A, B and C. While, restaurants A and B allow smoking, restaurant C does not provide a designated smoking area. Therefore, it is intuitive that (in this context) restaurants A and B are semantically closer to each other than to the restaurant C.

The second assumption means that relations are weighted according to their semantic importance. Let us now consider two aspects of a restaurant: availability of *live music* and type of *cuisine*. In general, we can assume that the cuisine is more important than the presence of live music. Henceforth, assuming that a tourist likes live music, we still can assume that he will probably choose a restaurant serving his favorite food, while without live music; rather than a restaurant, which serves the food that he does not like, but where he can listen to live music. Of course, the tourist can do otherwise (e.g. when the live music is his favorite blues band), but this still illustrates difference in importance of specific features of travel entities. To express such differences within the actual system, each relation is described by a weight representing its importance. Such weight can be also referred as a relevance value because it states how relevant to each other are the two objects connected by a given relation.

The third assumption states that connections (of the same type) between two individual objects can have different weights. This situation can be illustrated by the following observation: both, an "upper class" Chinese restaurant and a "low-quality" Chinese fast-food, serve meals demarcated as *Chinese cuisine*. However, the quality of meals in these restaurants differs significantly. Note that the *cuisine* relation has the same importance for both restaurants (assumption two). Thus, an additional weight has to be introduced, which describes user preferences with respect to an individual information object (e.g. one tourist may not care what is the type of the restaurant, while another may not want to eat in fast-food places).

6.3. Relevance calculation process. Based on these assumptions, and following the description found in [31], we can now present the the core component of the algorithm finding semantic closeness between two objects. Let us start from defining the *matching criterion*, which is an ordered quadruple $< x, q, a, g >$, where:

x is a source object,

q is a query which defines a subset of objects that are considered potentially relevant; these objects will be matched against the source object x,

$a \in (0; \infty)$ specifies the threshold of closeness between objects to be judged actually relevant to each other,

g is a sub-query which is used to optimize the matching process by reducing the number of considered nodes.

Calculation of the semantic closeness proceeds in two stages (1) Graph Generation, and (2) Relevance Calculation. In the first stage, an ontology is used to create a graph representing connections between the user (profile) and the travel entities. This graph is used, in the second phase, to find all direct and indirect relations between the source and the sink individuals. Therefore, the second stage has as its input the graph corresponding to the ontology, the source object and a set of target entities. Here, the algorithm calculates semantic closeness between the source and each target object separately. Let us briefly discus each stage (for the detailed description, see [31]).

6.3.1. Graph Generation. A graph generated in the first stage is called a *Relevance Graph*. The Relevance Graph is a directed label graph $G = (V, E)$, where:

- V is a set of nodes (a set of individual information objects),
- E is a set of edges (a set of relations).

Observe that the structure of the graph does not restrict the number of edges between two adjacent nodes. Therefore, if two information objects are connected by multiple relations, the corresponding two nodes (in the relevance graph) will also be connected by the same number of edges. Moreover, the relevance graph can contain cycles.

To optimize the node generation method, in the Travel Support System, a relevance graph is to be built only from the information objects, which are relevant with respect to the contextual information. This means that the algorithm will process only a small part of the ontology. Specifically, in the current implementation, the algorithm does not need to include information objects exceeding a certain distance from the user (which are filtered-out first). For example, a maximum distance can be set to 2 km for walking and 50 km for a car drive.

Edges in the relevance graph are generated based on the relations between objects in the ontology. Namely, each edge is defined as $e \in E = (x, y, distance, weight)$, where

- $x \in V$ is the tail node of the edge e,
- $y \in V$ is the head node of the edge e,
- $distance \in N$ is the conceptual distance value (see, the second assumption),
- $weight \in (0; 1) \in R$ is the individual weight value (see, the third assumption).

Finally, the distance between the two adjacent nodes is expressed by the formula

$$Distance = \frac{1}{Relevance}, \tag{6.1}$$

where the definition of the *Relevance* is based on how relevant are the two objects connected by a relation instantiated through the given edge. Note that, these values can be determined by a domain expert. Namely, a restaurant reviewer (the domain expert in the subject of restaurants) can adjust relevance values in such a way that the "cuisine" relation is more important than the "parking" relation, by assigning them the relevance values equal to 0.7 and 0.2 respectively. In practice, when the expert is not involved, all distance values should be initialized to a single value (e.g. distance = 1). In such case, all calculations will be performed based solely on the user profile (and the structure of the ontology). For the non-adjacent nodes, the distance value is going to be calculated in a way presented in the next section.

6.3.2. Relevance Calculation. The relevance calculation starts from "simplifying" structure of edges in the relevance graph. Here, we distinguish two operations: edge scaling and edge merging. Edge scaling individualizes each edge's distance by multiplying its relevance value by its weight. This operation is a consequence of the third assumption. Equation 6.2 presents a formula used to calculate a scaled relevance value for an edge $e = (x, y, Distance, weight) \in E$.

$$\begin{aligned} NewRelevance &= oldRelevance \times weight \\ NewDistance &= \tfrac{1}{NewRelevance} \end{aligned} \tag{6.2}$$

Since the structure of the relevance graph is restricted only by the ontology that it represents, there can be multiple edges between two adjacent nodes. The original publication suggested applying the following edge merging formula:

$$NewDistance = (\sum_{i=1}^{n} \frac{1}{Distance_i})^{-1} \tag{6.3}$$

After these operations, nodes in the Relevance Graph are connected by a single weighted edge. This edge represents semantic closeness of the two adjacent nodes (with respect to a given property).

Let us now consider the semantic closeness of non-adjacent nodes. First of all, the algorithm finds all valid paths between two non-adjacent nodes (the source node and one of the target nodes). A path is valid if and only if it is a simple path (namely, it does not contains repeating nodes). If there is no valid path between two non-adjacent nodes, it means that they are not related, hence their relevance is equal to 0 (their distance is equal to infinity). If there exists one or more valid paths, first, the algorithm calculates length of each of them by applying the following formula:

$$Distance_P = \sum_{i=1}^{n-1} i \times Distance_{a_i a_{i+1}} \tag{6.4}$$

Next, the algorithm calculates the cumulative relevance of all paths between the source node and the target node. Specifically, for n paths $P_1, P_2, \ldots, P_n$ from $x \in V$ to $y \in V$ the relevance value $Relevance_{xy}$ is defined as follows

$$Relevance_x y = \sum_{i=1}^{k} \frac{1}{Distance_{P_k}} \tag{6.5}$$

The $Relevance_x y$ represents the semantic closeness of two objects. This value is then to be used to select and rank objects to be recommended to the user.

7. Implementing the Rhee-Ganzha algorithm. The critical part of the Rhee-Ganzha algorithm is to determine all simple paths between two nodes (process of scaling and merging multiple path can be achieved by applying equations 6.2 and 6.3). In the original work [31] no details concerning possible / actual implementation were provided. Therefore, after some investigations, we have decided use the Yens algorithm [33] for finding k-shortest simple paths as the centerpiece of our implementation.

7.1. Yens algorithm. Yen's algorithm [33] is a deviation algorithm that finds k single-source shortest loopless paths in a graph with non-negative edge weights, by deviating the initial one found by another algorithm (for example, the Dijkstra's algorithm). Let us consider the k-th shortest path $p_k = \{{v_1}^k, {v_2}^k, \ldots, {v_n}^k\}$. In order to find the $(k+1)$-th shortest path, the algorithm analyzes every node ${v_i}^k$ in p_k and computes the shortest loopless path p, which deviates from the p_k at this node. The loopless path p_k is said to be a parent of p, and ${v_i}^k$ is its deviation node. To avoid loops during the calculation of a new path, the algorithm should remove all sub-paths between the source node and a deviation node ${v_i}^k$. Therefore, all such nodes are temporarily removed and the algorithm calculates a new path in the resulting graph.

Our implementation of the Yen's algorithm utilizes its new implementation, described in [30]. This implementation was proved to be generally more efficient than the other implementations (the publication covers the straightforward implementation and the implementation proposed by Perko). Specifically, the new implementation replaces the node deletion with its reinsertion. This allows to solve the problem faster, by labeling and correcting labels of some nodes. In general, the number of corrected nodes is assumed to be smaller than the total number of nodes in a graph. The tests performed by the authors of the article seem to prove this hypothesis. However, let us make it clear that, in the worst case scenario, the modified algorithm performs its computation in $O(Kn(m + n \log n))$, where K is the number of paths, n is the number of nodes and m is the number of edges. This time complexity is also the worst-case time complexity of the straightforward implementation.

Let us now describe the implementation of the Yen's algorithm that was used in our system. A shortest path tree rooted at a vertex x is a spanning tree T_x of a graph G. If x is the terminal node, then T_x represents the tree of the shortest paths from every node to x. Otherwise, T_x represents the tree of the shortest paths from x to every node. A loopless path from $v \in V$ to x in T_x is denoted by $T_x(v)$. The cost of $T_x(v)$ is denoted by π_v and is also used as a label of v. Taking this into account, the Yen's algorithm can be summarized as follows.

Yens Algorithm

Input:

1. G - a graph with the source and the target nodes
2. K - the number of shortest path to be found

Output: K the shortest paths

initialize an array *deviation*, which stores derivation nodes for all shortest paths
p = shortest path from the source to the target nodes in the graph G found by another algorithm (e.g. the Dijkstra's algorithm, see below)
$deviation[p] = s$
$X = \{p\}$
$k = 0$
while $X \neq \emptyset$ and $k < K$ **do**
  $k = k + 1$
  p_k = shortest, loopless path in G
  $X = X \setminus \{p_k\}$
  $\pi_v = \infty$, for any $v \in V$
  remove loopless path p_k, except target node t, from the graph G
  remove edges $(deviation(p_k), v), v \in V$ of $p_1, p_2, ..., p_k$
  T_t = shortest tree rooted at t
  for $v_i^k \in \{v_{l_k}^k, ..., d(p_k)\}$ **do**
    restore node v_i^k in the graph
    calculate $\pi_{v_i^k}$ (label) using forward star form
    if $\pi_{v_i^k}$ is defined **then**
      correct labels of v_i^k successors using backward star form
      $p = sub(s, v_i^k) \cdot T_t(v_i^k)$
      $deviation(p) = v_i^k$
      $X = X \cup \{p\}$
    end if
    restore (v_i^k, v_{i+1}^k) in the graph
    if $\pi_{v_i^k} > \pi_{v_{i+1}^k} + cost_{v_i^k, v_{i+1}^k}$ **then**
      $\pi_{v_i^k} = \pi_{v_{i+1}^k} + cost_{v_i^k, v_{i+1}^k}$
      correct labels of v_i^k successors using backward star form
    end if
  end for
  Restore the graph G
end while

Forward star formation
Correct labels of x successors
Input: a graph G

for edge $e = (i, j) \in E$ **do**
  if $\pi_i > \pi_j + cost_{i,j}$ **then**
    $\pi_i = \pi_j + cost_{i,j}$
  end if
end for

Backward star formation
Correct labels of x successors
Input: a graph G and a vertex x

$list = \{x\}$
repeat
  i = element of $list$
  $list = list \setminus \{i\}$
  for edge $e = (i, j) \in E$ **do**

if $\pi_i > \pi_j + cost_{i,j}$ **then**
$\pi_i = \pi_j + cost_{i,j}$
$list = list \cap \{j\}$
end if
end for
until $list \neq \emptyset$

In each iteration, the Yen's algorithm deviates from the previously found path to generate the new one. During this process, starting from the source node as a deviation node, the Yen algorithm calls another algorithm, which finds the shortest loopless path from the deviation node to the target. Based on the suggestion found in [33], we have decided to use the Dijkstra's algorithm to implement this step.

7.2. Dijkstras algorithm. The Dijkstras algorithm [26] is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path weights. It is also used to produce a shortest path tree. For a given source node in the graph, the algorithm finds a path with the lowest cost (the shortest path). Such path is used by the Yen's algorithm to generate the remaining part from the deviation node to the target. The algorithm proceeds as follows:

Dijkstras algorithm
Constructs an array of nodes which can be used to reconstruct the shortest path
Input: a graph G, the *source* and *target* nodes
Output: an array of nodes which can be used to reconstruct the shortest path

initialize an array *distance*, which stores shortest distances between the *source* and other nodes in G
initialize an array *previous*, which stores previous nodes in optimal path
for each vertex v in the graph G **do**
distance from the source to v (denoted by $dist[v]$) is equal to infinity
the previous node in the optimal path from the source to v (denoted by $previous[v]$) is undefined
end for
$distance[source]$ is equal to 0
initialize a priority queue Q, which contains all the nodes
while Q is not empty **do**
let u be a vertex in Q with the smallest distance from the source
remove u from Q
if $distance[u]$ is equal to infinity **then**
break
end if
for each neighbour v of u **do**
calculate an alternative distance $alt = distance[u] + distance(u, v)$
if alt is smaller than $distance[v]$ **then**
$distance[v] = alt$
$previous[v] = u$
$reordervinQ$
end if
end for
end while
return *previous*

The shortest path can be reconstructed by executing the following algorithm:
Reconstruct
Reconstructs the shortest path from an array of nodes.
Input: the previous array of nodes constructed by Dijkstras algorithm, the target node
Output: a sequence containing the shortest path

let S be an empty sequence
u is equal to the target

while *previous*[u] is defined **do**
  insert u at the beginning of S
  $u = previous[u]$
end while
return S

This algorithm can yield different performance depending on the implementation of the priority queue. Keeping in mind that the algorithm may be used with graphs built on the basis of large ontologies, the Dijkstra algorithm should operate in the shortest possible time. Therefore, to achieve the lowest time complexity, the decision was made to utilize the Fibonacci heap. This improves the time complexity of the algorithm to $O(|E| + |V| \log |V|)$, where E is a set of edges, V is a set of vertices and $|.|$ denotes the size of a set.

7.3. Fibonacci heap. The Fibonacci heap, used the Dijkstra's algorithm, is implemented according to [27]. A Fibonacci heap is a collection of item-disjoint heap-ordered trees. A heap-ordered tree is a rooted tree containing a set of items arranged in the heap order. Namely, if x is a node, then its key is no less than the key of the parent (provided that x has a parent). Thus, the root contains an item with the smallest key. There is no explicit constraint on the number or structure of the trees. However, the number of children of a node represents its rank. The rank of a node with n descendants is $O(\log n)$. The heap-ordered trees can perform an operation called linking. The linking operation combines two item-disjoint trees into one.

The heap is accessed by a pointer to its root, which contains an item with the smallest key. We can also call this root the minimum node. If the minimum node is undefined, the heap is empty. Each node contains a pointer to its parent, another pointer to one of its children and its rank. The children of each node are doubly linked in a circular list. This helps the heap maintain the low cost of its operations. For example, the double linking between a root and its children makes removing elements possible in $O(1)$. Similarly, the circular linking between children makes concatenation of two such lists possible in $O(1)$.

Since the Fibonacci heap will be used as a priority queue, the most important operations are the *delete_min* operation and the *decrease_key* operation. The *decrease_key* operation starts by subtracting a given number from the *key* of an item i in a heap h. Secondly, the algorithm finds a node x containing i and cuts the edge joining x to its parent (it also decreases the rank of the parent). As a result, the algorithm creates a new sub-tree rooted at x. If the new key of i is smaller than the key of the minimum node then the algorithm redefines the minimum node to be x. The complexity of the *decrease_key* operation is $O(1)$ (this operation assumes that the position of i in h is known).

The *delete* operation is similar to the *decrease_key* operation. First, the algorithm finds the node x containing i and cuts the edge with its parent node. Second, it concatenates the list of children of x with the list of roots and destroys x. The complexity of the delete operation is $O(1)$ (this operation assumes, again, that the position of i in h is known).

The *delete_min* operation requires finding pairs of tree roots of the same rank to link. This is achieved by using an array indexed by ranks. After deleting the minimum node and forming a list of new tree roots, the algorithm inserts the roots, one by one, into the array. If the root is inserted into an array position which is already occupied, then the algorithm performs the linking operation on the roots in conflict. After successful linking, the resulting tree is inserted to the next higher position. The *delete_min* operation ends when all the roots are stored in the array and its amortized time is equal to $O(\log n)$.

8. Matching Process Examples. Let us now illustrate the matching process on the basis of the two use case scenarios presented in Section 3. In both cases, the algorithm utilizes the Restaurant ontology described in Section 5. In the first example, we will perform relevance calculations based on the following matching criteria from the first use case scenario – "finding a vegetarian restaurant."

1. $x = User\_Profile\_Restaurant$
2. $q =$
   $PREFIX onto : \langle http://localhost:9000/assets/Ontologies/Restaurant/Restaurant\rangle$
   $SELECT ?restaurant$
   $WHERE ?restaurant is a onto : Restaurant.$
3. $a = \frac{3}{5}$

4. $g = [lat = 22.2, long = 44, dist = 2000]$:
   lat = user's latitude
   $long$ = user's longitude
   $dist$ = the maximum distance (in meters) between the source (the user's current position) and a target object

Note that the source object x, *User_ Profile_ Restaurant*, does not exist in the ontology. It is an information object created on the basis of the user profile. Recall, the currently we apply a simplified representation of user profiles. For example, the source object corresponding to the user profile presented in section 5 has the following relations:

1. *res:User_ Profile_ Restaurant res:cuisine cui:Vegetarian*
2. *res:User_ Profile_ Restaurant res:cuisine cui:Indian*
3. *res:User_ Profile_ Restaurant res:cuisine cui:SteakHouses*
4. *res:User_ Profile_ Restaurant res:smoking acs:SmokingArea*
5. *res:User_ Profile_ Restaurant res:dress drs:Casual*
6. *res:User_ Profile_ Restaurant res:restaurantCategory rcc:FastFoodRestaurant*
7. *res:User_Profile_Restaurant res:restaurantCategory rcc:CasualRestaurant*

Then, to determine a set of possibly relevant target objects, the matching criteria defines the q query and g sub-query (see, section 6.3). In our example, the user is interested in restaurants. Hence, the q query accepts all individuals, which are instances of the *Restaurant* class. However, the q query does not determine, which information objects are located within a certain distance from the user (here, a sample value $dist = 2000$). Therefore, the g query removes all places located further than a given distance. By combining these two queries, the algorithm is able to obtain all restaurants located not further than 2 km from the user. In what follows, relevance calculations will be performed for sample restaurants *JoesSteakHouse*, *ThaiIndiaRestaurant* and *LuckyChefRestaurant*. Therefore, the algorithm calculates the semantic closeness between the source (Jack's profile) and the three target objects found in the previous step. The matching process involves:

1. source instance URI = *User_Profile_Restaurant*
2. target objects URI's = [*Joe's Steak House*, *ThaiIndiaRestaurant*, *LuckyChef Restaurant*]
3. relevance threshold: $a = \frac{3}{5}$

The relevance calculation starts from creating a *Relevance Graph* (described in section 6.3), which corresponds to the Restaurant Ontology depicted in figure 5.1. Since the resulting graph does not have multiple relations between any two adjacent nodes, the system skips the edge merging procedure. Now, let us recall that different features described by the Restaurant ontology can have different individual weights stored in the user profile. For example, *Indian* and *SteakHouses* cuisines have their individual weights equal to 0.8 and 0.45 respectively. Therefore, the algorithm needs to scale the edges by applying equation 6.2. Recalling our discussion in section 6.3.1, currently all edges have their initial distances equal to $d = 1$. Therefore, the calculations proceed as follows:

$$distance_{Vegetarian} = (\frac{1}{d_{cuisine}} \times w_{Vegetarian})^{-1} = (1 \times 0.91)^{-}1 = 1.1$$

$$distance_{Indian} = (\frac{1}{d_{cuisine}} \times w_{Indian})^{-1} = (1 \times 0.65)^{-}1 = 1.54$$

$$distance_{SteakHouses} = (\frac{1}{d_{cuisine}} \times w_{SteakHouses})^{-1} = (1 \times 0.04)^{-}1 = 25$$

Recall that if the user profile does not specify the weight for an information object, then a default weight is applied (here, $w = 0.5$). After creating the appropriate *Relevance Graph*, the algorithm calculates relevance values establishing relations between the source (*User_Profile_Restaurant*) and the target objects (*ThaiIndiaRestaurant*, *Joe's Steak House* and *LuckyChefRestaurant*):

ThaiIndiaRestaurant

The system utilizes the Yen's algorithm (described in section 7.1) to find all simple paths. Based on the *Relevance Graph*, we can define four paths from the *User_Profile_Restaurant* to the *ThaiIndiaRestaurantRestaurant*:

Path $1: User\_Profile\_Restaurant \rightarrow Indian \rightarrow ThaiIndiaRestaurant$
Path $2: User\_Profile\_Restaurant \rightarrow SmokingArea \rightarrow ThaiIndiaRestaurant$
Path $3: User\_Profile\_Restaurant \rightarrow Casual \rightarrow ThaiIndiaRestaurant$
Path $4: User\_Profile\_Restaurant \rightarrow CasualRestaurant \rightarrow ThaiIndiaRestaurant$

By applying equation 6.4, the respective distance values are:

$D_{Path1} = 1.1 + (2 \times 1.1) = 3.75$
$D_{Path2} = 3.33 + (2 \times 3.33) = 10.0$
$D_{Path3} = 1.52 + (2 \times 1.52) = 4.56$
$D_{Path4} = 1.43 + (2 \times 1.43) = 4.29$

Finally, the algorithm obtains the total relevance value, from equation 6.5:

$$Rel_{ThaiIndiaRestaurant} = \frac{1}{3.75} + \frac{1}{10.0} + \frac{1}{4.56} + \frac{1}{4.29} = 0.8191$$

JoesSteakHouseRestaurant

All simple paths between the *User_Profile_Restaurant* and the *JoesSteakHouseRestaurant* are:

Path $1: User\_Profile\_Restaurant \rightarrow SteakHouse \rightarrow JoesSteakHouseRestaurant$
Path $2: User\_Profile\_Restaurant \rightarrow SmokingArea \rightarrow JoesSteakHouseRestaurant$
Path $3: User\_Profile\_Restaurant \rightarrow Casual \rightarrow JoesSteakHouseRestaurant$
Path $4: User\_Profile\_Restaurant \rightarrow CasualRestaurant \rightarrow JoesSteakHouseRestaurant$

By applying equation 6.4 we obtain:

$D_{Path1} = 25 + (2 \times 25) = 75.0$
$D_{Path2} = 3.33 + (2 \times 3.33) = 10.0$
$D_{Path3} = 1.52 + (2 \times 1.52) = 4.56$
$D_{Path4} = 1.43 + (2 \times 1.43) = 4.29$

Next, by applying equation 6.5 we calculate the total semantic closeness as:

$$Rel_{JoesSteakHouseRestaurant} = \frac{1}{75.0} + \frac{1}{10.0} + \frac{1}{4.56} + \frac{1}{4.29} = 0.5657$$

LuckyChefRestaurant

All simple paths between the *User_Profile_Restaurant* and the *LuckyChefRestaurant* are:

Path $1: User\_Profile\_Restaurant \rightarrow Vegetarian \rightarrow LuckyChefRestaurant$
Path $2: User\_Profile\_Restaurant \rightarrow Casual \rightarrow LuckyChefRestaurant$
Path $3: User\_Profile\_Restaurant \rightarrow CasualRestaurant \rightarrow LuckyChefRestaurant$

By applying equation 6.4 we obtain:

$D_{Path1} = 1.1 + (2 \times 1.1) = 6.69$
$D_{Path3} = 1.52 + (2 \times 1.52) = 4.56$
$D_{Path4} = 1.43 + (2 \times 1.43) = 4.29$

Next, by applying equation 6.5 we establish the semantic closeness to be equal to:

$$Rel_{LuckyChefRestaurant} = \frac{1}{3.3} + \frac{1}{4.56} + \frac{1}{4.29} = 0.7554$$

Based on these results, and the proposed threshold value $a \geq \frac{3}{5}$, the *LuckyChefRestaurant* (0.7554) and the *ThaiIndiaRestaurant* (0.8191) can be recommended to the user. Since the relevance value of the *JoesSteakHouseRestaurant* (0.5657) is below the threshold, the algorithm will not include this recommendation in the results. The ranking of the recommended restaurants is also possible and will place the *ThaiIndiaRestaurant* as the overall winner.

In the second example, we assume that Jack would like to invite Jill to a restaurant (second use case scenario). Here, the Travel Support System can prepare a list of the most suitable restaurants. To simplify the calculations, let us assume that Jack will invite Jill to one of the restaurants for which we have just completed calculation of their semantic closeness to his preferences. Therefore, we know that the two restaurants that are the suitable for him are: *LuckyChefRestaurant* and *ThaiIndiaRestaurant*. Thus, his system starts negotiations with Jill's instance of the Travel Support System by sending a message containing these two restaurants as an initial proposal. Based on the description of the problem, the relevance calculations seen from Jill's perspective

can be expressed as:

1. source instance URI = *JillsRestaurant*
2. target objects URI's = [*LuckyChefRestaurant*, *ThaiIndiaRestaurant*]
3. relevance threshold: $a = \frac{3}{5}$

Let us now assume that Jill's user profile is as follows:

1. (*cur:Vegetarian*, 0.91)
2. (*cur:Indian*, 0.65)
3. (*cur:SteakHouses*, 0.04)
4. (*asc:SmokingArea*, 0.3)
5. (*drs:Casual*, 0.66)
6. (*rcc:FastFoodRestaurant*, 0.1)
7. (*rcc:CasualRestaurant*, 0.7)

Then, the semantic closeness information object corresponding to her profile and the restaurants proposed by Jack is:

ThaiIndiaRestaurant

All simple paths between *JillsRestaurant* and the *ThaiIndiaRestaurant* are:

Path 1 : $JillsRestaurant \rightarrow Indian \rightarrow ThaiIndiaRestaurant$
Path 2 : $JillsRestaurant \rightarrow SmokingArea \rightarrow ThaiIndiaRestaurant$
Path 3 : $JillsRestaurant \rightarrow Casual \rightarrow ThaiIndiaRestaurant$
Path 4 : $JillsRestaurant \rightarrow CasualRestaurant \rightarrow ThaiIndiaRestaurant$

By applying equation 6.4, the respective distance values are:

$D_{Path1} = 1.18 + (2 \times 1.18) = 3.54$
$D_{Path2} = 1.67 + (2 \times 1.67) = 5.00$
$D_{Path3} = 3.34 + (2 \times 3.34) = 10.0$
$D_{Path4} = 1.43 + (2 \times 1.43) = 4.29$

Finally, the algorithm obtains the total semantic closeness value by utilizing equation 6.5:

$$Rel_{ThaiIndiaRestaurant} = \frac{1}{3.54} + \frac{1}{5.0} + \frac{1}{10.0} + \frac{1}{4.29} = 0.7156$$

LuckyChefRestaurant

All simple paths between *JillsRestaurant* and the *LuckyChefRestaurant* are:

Path 1 : $JillsRestaurant \rightarrow Vegetarian \rightarrow LuckyChefRestaurant$
Path 3 : $JillsRestaurant \rightarrow Casual \rightarrow LuckyChefRestaurant$
Path 4 : $JillsRestaurant \rightarrow CasualRestaurant \rightarrow LuckyChefRestaurant$

By applying equation 6.4 we obtain:

$D_{Path1} = 2 + (2 \times 2) = 6.0$
$D_{Path2} = 3.34 + (2 \times 3.34) = 10.0$
$D_{Path3} = 1.43 + (2 \times 1.43) = 4.29$

Finally, by applying equation 6.5 we can calculate the total semantic closeness:

$$Rel_{LuckyChefRestaurant} = \frac{1}{6.0} + \frac{1}{10.0} + \frac{1}{4.29} = 0.4998$$

Based on these results, and the proposed matching criteria ($a \geq \frac{3}{5}$), where a is the relevance threshold, only the *ThaiIndiaRestaurant* will be recommended to Jill. Because the relevance value of *LuckyChefRestaurant* is below the threshold, the algorithm will not include this recommendation in the results. Therefore, Jack will be informed that he should propose to Jill the *ThaiIndiaRestaurant* restaurant (since it matches their preferences at least at the minimal level).

9. Concluding remarks. The aim of this paper was twofold. First we have discussed the main assumptions leading to the new version of an agent-semantic travel support system. Second, we have shown how the Rhee-Ganhza algorithm can be used to establish semantic closeness between the user profile and the restaurant objects. The system has been implemented and works on Android-based mobile devices. As a matter of fact, the results illustrating its work, presented in section 8, have been prepared on the basis of the working system. The next step will be to extend the travel ontology to be able to test the proposed approach on more complex ontologies, while asking more sophisticated queries.

REFERENCES

[1] *A personal agent application for the semantic web*. http://www.csee.umbc.edu/~finin//papers/aaaiss03.pdf.
[2] *Applying semantic search and ontology to the travel industry*. http://www.tnooz.com/2011/05/20/news/applying-semantic-search-and-ontology-to-the-travel-industry/.
[3] *Booking.com*. http://www.booking.com/.
[4] *Booking.com mobile application*. http://www.booking.com/general.html?sid=1b58219367c2cba11085678296d6347e;dcid=1;tmpl=docs%2Fiphone_landing&lang=en-gb.
[5] *Facebook*. https://www.facebook.com/.
[6] *Google Now*. http://www.google.com/landing/now/.
[7] *iPic Entertainment*. https://www.ipictheaters.com/default.aspx.
[8] *iPic Theaters mobile application*. https://play.google.com/store/apps/details?id=nz.co.vista.android.movie.ipic&feature=search_result#?t=W251bGwsMSwxLDEsIm56LmNvLnZpc3RhLmFuZHJvaWQubW92aWUuaXBpYyJd.
[9] *Open Directory Project*. http://www.dmoz.org/.
[10] *Openrice*. http://www.openrice.com/.
[11] *OpenRice mobile application*. https://play.google.com/store/apps/details?id=com.openrice.android&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS5vcGVucmljZS5hbmRyb2lkIl0.
[12] *Pellucid (A Platform for Organisationally Mobile Public Employees)*. http://www.softeco.it/en/pellucid1.html.
[13] *Pellucid Project*. http://cordis.europa.eu/search/index.cfm?fuseaction=proj.document&PJ_RCN=5519298.
[14] Krawczyk K., Majewska M., Dziewierz M., Sota R., Balogh Z., Kitowski J., Lambert S., *Reuse of Organisational Experience Harnessing Software Agents*, Computational Science - ICCS 2004, Springer, 2004, pp. 583–590.
[15] *Resource Description Framework*. http://www.w3.org/RDF/.
[16] *Semantic Web*. http://www.w3.org/standards/semanticweb/.
[17] *Smart travel service advisor using Semantic Web and Agent Technology*. http://eprints.soton.ac.uk/266943/1/Travel_Advisor_paper.pdf.
[18] *Travel Support System publications*. http://www.ibspan.waw.pl/~paprzyck/mp/cvr/research/agents_TSS.html.
[19] *Travel Support System Sourceforge*. http://e-travel.sourceforge.net/.
[20] *TripAdvisor*. http://www.tripadvisor.com/.
[21] *Twitter*. https://twitter.com/.
[22] *Virtual consultant for public services, presentation in Polish)*. http://archiwum.spnt.pl/konwent2009/files/pdf/abg.pdf.
[23] *Virtual consultant for public services)*. http://www.um.radlin.pl/943,aaa.html.
[24] Maes P., *Agents that reduce work and information overload*, in Communications of the ACM Volume 37 Issue 7, 1994, p. 30-40.
[25] *Yelp!* http://www.yelp.com/.
[26] R. R. L. S. C. Cormen Thomas H., Leiserson Charles E., 2011.
[27] T. R. E. Fredman M. L., *Fibonacci Heaps and Their Uses in Improved Network*, Journal of the ACM, 1987.
[28] P. M. G. M. Gawinecki M., Kruszyk M., *Pitfalls of agent system development on the basis of a travel support system*, in Proceedings of the BIS 2007 Conference, Springer, 2007, pp. 488–499.
[29] A. Nauli, *Using software agents to index data of an e-travel system*, Master thesis, Oklahoma State University, 2000.
[30] M. E. Pascoal M., *A new implementation of Yens ranking loopless paths algorithm*, 4QR - Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 2003.
[31] P. M.-W. S. M. F. G. G. M. P. M. Rhee S. K., Lee J., *Measuring semantic closeness of ontologically demarcated resources*, in Fundamenta Informaticae, 2009, pp. 395–418.
[32] S. B. Ricci F., Rokach L., *Introduction to Recommender Systems Handbook*, Springer, 2011.
[33] Y. J. Y., *An algorithm for finding shortest routes from all source nodes to a given destination in general networks*, in Quarterly of Applied Mathematics 27, 1970, p. 526530.

Edited by: Viorel Negru and Daniela Zaharie
Received: June 1, 2013
Accepted: Jul 15, 2013

DOI 10.12694/scpe.v14i2.845
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. 131–141. http://www.scpe.org

DESIGNING A SCALABLE SOCIAL E-COMMERCE APPLICATION

EUGENIO ZIMEO[1] AND GIANFRANCO OLIVA, FABIO BALDI, ALFONSO CARACCIOLO[2]

Abstract. eCommerce is gaining a momentum due to the wide diffusion of Web 2.0 technology. Social mining, recommenders and data semantics are moving the focus of eCommerce applications towards context-awareness and personalization. However, the design of these software systems needs specific architectures to support intelligent behaviors, still ensuring important non-functional properties, such as flexibility, efficiency and scalability. This paper proposes an architectural pattern that helps designers to easily identify the subsystems that characterize intelligent enterprise systems. By decoupling transactional behavior from batch processing, the pattern avoids the interference of knowledge extraction and reasoning processes with the state and the performance of the transactional subsystem, so improving scalability. The pattern has been experimented in eCommerce by designing an intelligent and scalable virtual mall.

Key words: Architectural Pattern, Software Systems Design, Scalability, Enterprise Systems, Intelligent Systems, eCommerce

1. Introduction. Enterprise systems represent today an important class of large-scale software that supports many fundamental processes of complex organizations, ranging from resource planning to business intelligence. In this class, eCommerce applications often groups many enterprise assets to offer an integrated and coherent view to merchants and customers for performing business actions.

The adoption of eCommerce as a prevalent channel for selling products is changing the market rules, since competition among vendors is being migrated on the Web. Reaching the widest set of potential customers with information about the commercial products of their interest is one of the challenges that characterize the new solutions and technologies for eCommerce applications.

In the evolution of the Web, Web 2.0 technology represents a milestone with its emphasis on supporting social collaboration and reasoning over data semantics. The new way of interaction is introducing a radical innovation in eCommerce [21], moving the focus towards context-awareness and personalization. If on the one hand, these innovative features improve product selling, they also significantly impact the way modern eCommerce applications are designed and implemented.

Traditional architectures exploited to design enterprise systems (in particular eCommerce applications) need to be revised in order to take into account the desired ability of these systems to "generate" knowledge while working, on the basis of several information sources that could be available as enterprise assets. These sources could be tightly related to the eCommerce application or belonging to different enterprise subsystems that support cross-business features; they can change over time or can be enriched with additional ones when new needs emerge.

Designing an architecture for eCommerce applications in this new scenario is not simple, since designers have to combine the expected intelligent behavior of the system with non-functional requirements, such as flexibility, efficiency and scalability. Separation of concerns, already applied to design complex architectures by separating orthogonal non-functional aspects in insulated modules, could be a viable approach also to design flexible and scalable intelligent enterprise systems.

This paper presents an architectural pattern that helps satisfying both functional and non-functional scalability during the design of intelligent enterprise systems. The pattern decouples the transactional behavior from batch processing, in order to avoid dangerous interference of knowledge extraction and reasoning processes with the state and the performance of the transactional subsystem.

The pattern is applied to the design of a complex eCommerce application, which virtualizes a *mall* composed of a variable number of eShops. The pattern is adopted as a key step of the ADD - *Attribute Driven Design* method [18], which is exploited as a reference process model to design the whole system. The paper shows the effects of the pattern to simplify the comprehension of the system and to improve its design.

The rest of the paper is organized as follows. Section 2 discusses some common architectural patterns exploited to design eCommerce applications and the role of patterns in designing intelligent systems. Section 3

[1]University of Sannio, Dep. of Engineering, 82100 - Benevento, ITALY (zimeo@unisannio.it)
[2]Poste Italiane S.p.A., TI - FSTI - Centro Ricerca, 80133 - Napoli, ITALY (OLIVAG11@posteitaliane.it, BALDIFA3@posteitaliane.it, CARACC52@posteitaliane.it)

presents the *Inference Pattern* proposed. Section 4 discusses the application of the pattern in designing a real system. Finally, Section 5 concludes the paper.

2. Background and related work. Architectural patterns, design patterns and idioms constitute an extensible knowledge base that helps designers to take proper decisions in short times, when they design complex software systems, by reusing solutions already experimented in similar application contexts.

Several architectural patterns have been proposed in the literature for different classes of software systems. In this section, we mainly refer to the patterns that have been successfully exploited for designing enterprise systems and we introduce some recently proposed patterns for intelligent systems, which inspired the one presented and discussed in this paper.

The Treasury Architecture Development Guidance (TADG) [3] proposes some general architectural patterns that can be also exploited to design enterprise systems: *Client-Proxy Server*, *Customer Support*, *Reactor*, *Replicated Servers*, *Layered Architecture*, *Pipe and Filter*, *Subsystem Interface*.

An interesting proposal comes from IBM [1, 2]. It identifies several patterns at different abstraction levels to guide the design of e-Business applications, a specific class of enterprise systems including eCommerce. In this vision, user requirements drive the selection of one or more Business patterns among *Self-Service, Collaboration, Information Aggregation, and Extended Enterprise*.

The first one captures the business interaction between a user and a service, typically a simple web site. The second one enables the collaboration among users. The third one allows users to aggregate data to extract information that is useful for business purposes (e.g. business intelligence). Finally, the fourth one enables a business-to-business interaction to extend the functions of an enterprise with the ones provided by other networked enterprises (e.g. supply chain management).

The business patterns are then used to suggest the adoption of *application* and *integration* patterns, often called *architectural* patterns. The formers capture the functional and non-functional requirements of an application, by proposing the system decomposition in functional and logical subsystems. The latters suggest a way to integrate different subsystems: *Access Integration* patterns define a common entry point for accessing services whereas *Application Integration* patterns represent a way to integrate the flow of actions or the data owned by different subsystems. In the literature, a lot of integration patterns, mainly based on messaging, have been introduced to support Enterprise Application Integration (see [5] for a presentation of these patterns).

A widespread architectural pattern that satisfies self-service interaction model is *Model View Controller* (MVC) [6, 7]. This pattern, with its main variants (*Model View Presenter* [6] and *Presentation Abstraction Controller* [9] are the most known), suggests the organization of the presentation layer and its decoupling from the model of enterprise systems. However, additional patterns are needed to help the design of the other layers proposed by multi-layer and multi-tier decompositions [10]. The layered organization allows for a clear separation of presentation, application logic and data management that multi-tiered architectures exploit in order to partition the different functional components onto dedicated resources, so ensuring important non-functional attributes, such as security, scalability, efficiency and data persistence. In this context, enterprise patterns [4] assume an important role in guiding the detailed design of complex applications.

Service Oriented Architecture (SOA) is becoming the reference high-level architectural pattern to design flexible, reusable and dynamic enterprise and inter-enterprise systems, especially when asynchronous interactions, based on an *Enterprise Service Bus*, are exploited. In [11], the authors propose a mediator-based architecture that decouples service providers from consumers with the aim of binding services on demand on the basis of customers' preferences.

However, the "intelligent" dimension of enterprise systems needs more autonomous and proactive behaviors to satisfy users' needs. Autonomic computing [12] is emerging as a promising approach to include self-* properties in software design and workflow systems [22], whilst MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge) [13] is often used as a reference architectural pattern to design these software systems. This pattern allows for observing the state of a resource in order to intercept possible deviations from a desired behavior that can be controlled by planning adaptations. However, it represents only a starting point for intelligent enterprise systems, where specific patterns are needed to design systems with the ability of inferring knowledge from data generated during the execution.

The paper in [20] focuses on a knowledge management architecture to apply data mining to eCommerce

but it does not address the general architecture of the system. In [14], the authors discuss an architecture to efficiently perform OLAP on the data produced by customers through the interaction with an eCommerce site. Even if, the idea of closing the knowledge loop - users, application, analysis, application changes, users - is similar to the one proposed in this paper, the architecture does not provide users with sufficient hints about the identification of finer-grained subsystems.

In this paper, we present a new architectural pattern, called *Inference Pattern*, that provides software architects with a conceptual framework for designing the business logic of intelligent enterprise systems.

3. Inference Pattern. The pattern will be described according to a typical schema for patterns.

Problem. We want to design a software system able to perform processing on its own data, produced by user interactions, with the aim of expliciting the implicit knowledge that may be useful to users and to the system itself.

Context. Interactive applications generate data from users interactions, which are compliant to models that are typically described through relational, ontological or object-oriented schemas. Data passed during interactions can be processed with the support of data coming from other sources to generate new information that can be useful to understand users' behaviors and preferences. This knowledge, often defined implicit or tacit, can be exploited to personalize the interaction between users and the system, since it eases the knowledge transfer from software to users, by reducing the effort to access data. This improves the effectiveness of the interaction with reference to the business objectives of the system. The knowledge acquired by the system can be in turn reused by other systems.

Solution. The system is decomposed in two logically separated subsystems: the one in charge of answering to user requests and the one responsible of collecting data and preprocessing them. The former becomes passive with reference to the latter that, on the other hand, generates the aggregated data that are useful to improve user interactions and the quality of the data hosted by the interactive subsystem. The logical separation promotes both the knowledge base extensibility, thank to the possibility of using additional systems as subject, and the continuous and concurrent processing with respect to the transactional activity generated by the interactive subsystem, so easing and optimizing the successive deployment.

Participants. In the following, a list of participating subsystems (see Fig. 3.1) is given with a brief description.

- Subject: is the observed subsystem from which implicit knowledge is extracted.
- *Knowledge extraction:* collects the data produced by the interactions between users and the subject and extracts the implicit knowledge.
- *Generation:* generates new knowledge for the subject, by using the implicit knowledge extracted by the Knowledge Extraction (KE) subsystems.
- *Decision Support:* it provides the user with a new knowledge extracted by the KE subsystem to support proper decisions.
- *Recommendation:* suggests information to users. It may use the Generation subsystem to ask for the generation of aggregated data or the Decision Support subsystem to recommend a choice to a user.
- *Personalization:* it personalizes the suggestions with respect to the specific user (or context in general). It is a sort of recommender supporting personalization.

Structural view. In Fig. 3.1, the structural view of the pattern is described by highlighting the participants introduced above and the *use* relationships among them. The red dotted line is a particular use dependency since it, differently from the other ones, works on the Subject by writing onto it. In addition, the view shows two subsystems of KE: *Management* and *Search*, which can be useful to configure the inference rules and to perform particular search operations on the collected data.

The KE subsystem can access the Subject by means of synchronous (dependency regards the external operations of the subject) or asynchronous (dependency is related to the events) interactions.

Related patterns. The proposed pattern is inspired by two known patterns: *Observer* [6] and *MAPE*. The former is a design behavioral pattern that introduces the concept of observing the events generated by a Subject (or Observable system) to drive the behavior of a multitude of Observers. The latter, on the other hand, changes the viewpoint with respect to the subject, which becomes a resource monitored by a manager that is able to analyze the state of the resource and to apply to it, if needed, some state changes (that in turn may produce

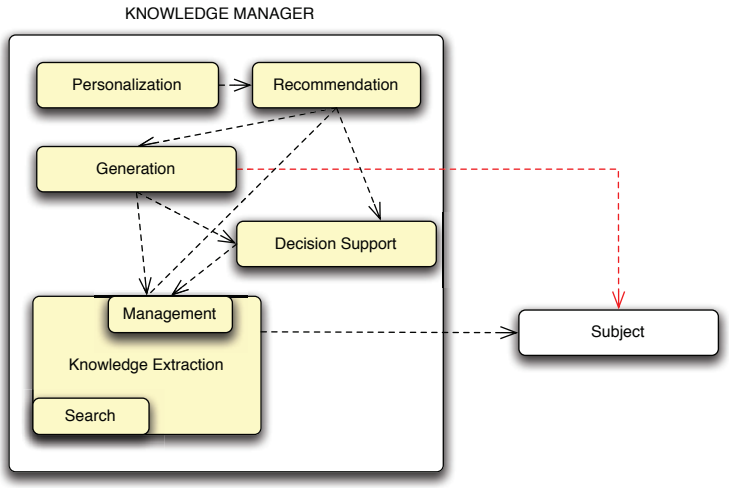

FIG. 3.1. *Structural view of the pattern*

behavioral changes).

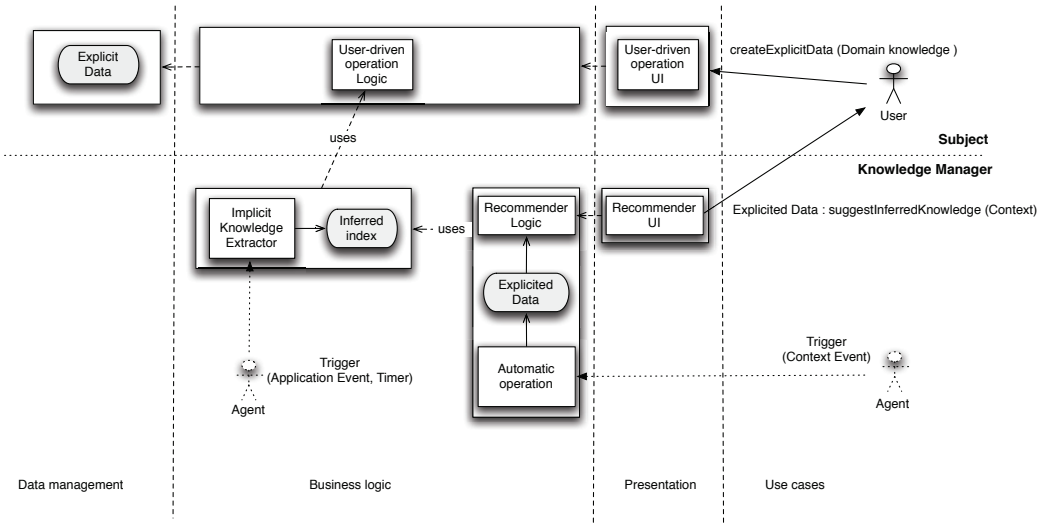

FIG. 3.2. *Partial application of the pattern in a multi-tier context*

The Inference Pattern, differently from MAPE, does not aim at conferring an autonomic behavior to the system but mainly at providing users with an additional knowledge inferred from the data collected from the subject. Therefore, even if, like MAPE, it creates a closed-loop system, the closure is mainly performed by users and, when possible, by the system itself through the writing operations (Generation) applied to the state of the subject. Typically, these operations do not change the behavior of the system, but they make it possible to retrieve further information by users, to feed the knowledge cycle.

Example. Fig. 3.2 shows an example of the pattern application in a typical case that combines interactive transactions derived from users (user-driven operations that store data coming from users' knowledge) and concurrent data extraction and elaborations of (automatic) recommendations to the same user. The (software) agents are in charge of performing the use cases that regard automatic operations. The results of these operations represent the inferred knowledge that is used to provide suggestions in a given context. The figure also shows

the separation of the subsystems in finer grained logical ones, useful to implement a layered architecture that can be easily transformed in a multi-tier one thanks to the allocation of the subsystems onto different resources.

4. Case study: the InViMall system. The Inference Pattern has been exploited to design a scalable eCommerce system, called *InViMall* (*In*telligent *Vi*rtual *Mall*), at Poste Italiane S.p.A.

InViMall is an electronic mall, based on e-community concepts, that enhances the shopping experience of users. It exposes several innovative features through a multichannel interface:

- *Personalized suggestions* [15]. They provide a personalized selection of products based on users interests, preferences, interaction history and the history of related users. Recommenders exploit the relationships between users and products to aid customers in selecting items from a set of choices, suggesting the products fitting their tastes. Users' profiles are either explicitly defined or inferred by analyzing their behaviors during the interaction with the system.
- *Automatic generation of personalized product bundles* [16]. They find the combinations of products that try to satisfy user preferences and requirements, guaranteeing, at the same time, the satisfaction of merchants needs, such as the minimization of the dead stocks.
- *Advanced marketing intelligence* [17]. They propose suggestions to merchants, based on the analysis of sales data from the entire system combined with the profiles of the registered users. They exploit also social mining techniques to create viral campaigns targeted to the most influential members within a community.
- *Faceted Navigation.* They provide customers and merchants with a semantic multi-dimensional search, which is able to reach the desired product by using different attributes as entry point in the catalogue.

TABLE 4.1
Functional areas and potential functional subsystems

Name	Description
eCommerce	eShops mgmt, multi-shop order mgmt, catalogues, shopping list
Model Manager	Semantic catalogue, product/customer relationships
Marketing	Marketing campaigns, analysis of sales and reporting, buyer groups
Social Network	Relationship among users, thematic groups and content sharing
Social Analyzer	Social mining: opinion leader and friendships identification
Selection and Bundle Generator	Prediction rating and automatic bundle generation
Other Systems	Interface to payment, logistics, shipping, and financial systems
Communication	Notifications, e-mails and messages exchange among users
Social Commerce	Products, eShop and bundle ratings, reviews and likes
Personalization	Products and bundle suggestions based on users preferences
Search	Shop and Mall catalogue search, faceted navigation

These functions have to be implemented taking into account typical non-functional requirements for eCommerce systems, such as multichannel access and interoperability with external systems. The multichannel access regards different kinds of devices (e.g. personal computers, mobile devices, kiosks) and communication channels (http, dedicated protocols). Interactions with external systems regard:

- (a) the real-time monitoring of the entire product delivery process across the country, by exploiting the Postal System Logistics platform of Poste Italiane;
- (b) payments, by using any of the several methods offered by Poste Italiane.

4.1. InviMall design. The design of InViMall was performed by following the *Attribute Driven Design* method proposed in [8]. According to this method, it is important to clearly define the requirements, both functional and non-functional with the aim of identifying the architectural drivers.

Therefore, the first phase of the process was the identification of scenarios and use cases that cover the innovative aspects of the system. Hence, some functional areas and possible subsystems were initially identified (see Table 4.1)

Due to the absence of a reference architectural pattern for the business logic of this kind of systems, the

next step was the identification of dependencies among subsystems, by exploiting use cases analysis, with the aim of building a *Dependency Structure Matrix* (DSM) [19]. It was useful for identifying and analyzing the mutual dependencies among the subsystems.

As Fig. 4.1 shows, reasoning only about the functional areas created a dependency graph with several mutual-dependencies, so generating a lot of coupling in the systems that could negatively impact flexibility, reusability, performance and scalability.

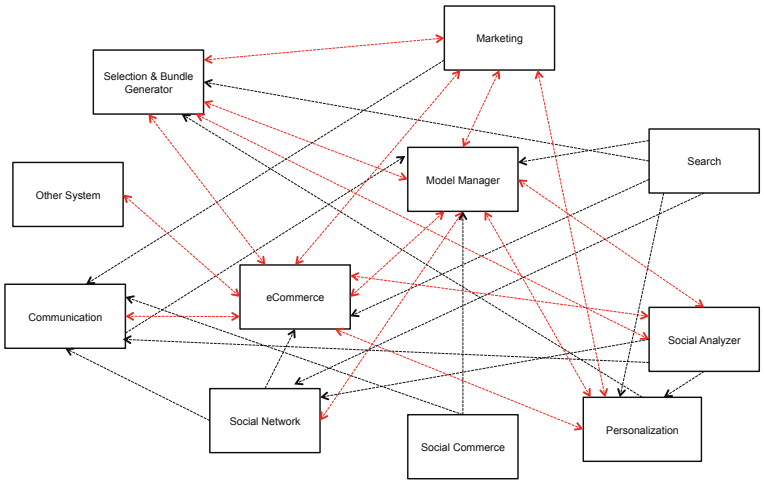

FIG. 4.1. *Preliminary decomposition of the InViMall system in subsystems*

4.2. Application of the Inference Pattern. The ADD method, to design complex software systems, suggests to identify the architectural drivers from the software requirements with the aim of selecting the proper architectural patterns to adopt as a reference guide for the whole design phase. This allows software architects to reuse the consolidated experience as an existing knowledge base to reduce the design effort.

In the case of complex and innovative software systems, existing patterns could not be sufficient to identify subsystems and their relationships, so limiting the benefits of the ADD method. We encountered this problem during the beginning of the design phase of InViMall, due to the lack of specialized patterns for designing the business logic of intelligent eCommerce applications. To overcome the problem, the architectural drivers of the InViMall system were generalized and an effort was done to abstract a solution by defining the architectural pattern presented in the paper.

The application of the pattern caused a complete refactoring of the design (see Fig. 4.2), introducing a strong reduction of mutual dependencies and a clear separation between the interactive and transactional subsystem (called *Mall*), and the batch subsystem (called *Intelligent Mall*).

The former groups the typical subsystems that characterize an electronic mall, extended with social network features. The latter hosts (i) a KE, which implements knowledge reasoning, predicted rating features and social mining, (ii) a decision support subsystem that enables merchant to configure and manage innovative marketing campaigns, (iii) a *Generation* subsystem that works onto the state of the Mall to store new explicit knowledge extracted by KE from the implicit one collected from several sources. It is worth to note that several features were tangled among different subsystems before applying the pattern, so generating mutual dependencies and coupling.

The two main subsystems derived from the decomposition have different characteristics that impact their implementation. The Mall is a typical transactional system and therefore it is characterized by concurrent accesses, data persistence and ACID transaction management. On the contrary, the Intelligent Mall is mainly a batch system using background processes to manipulate large amounts of data for analytical processing and to generate correlations among entities. Due to these peculiarities, non-relational persistence systems (e.g.

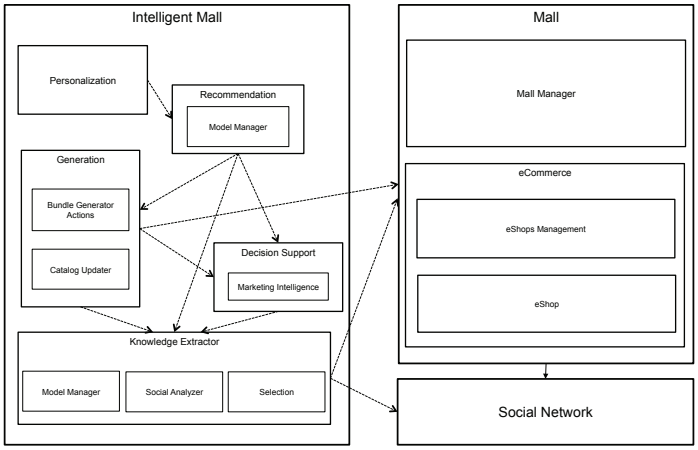

FIG. 4.2. *InViMall architecture after the application of the pattern*

columnar or graph-based DBs) could be exploited to improve the performance of the whole Knowledge Manager.

The derived architecture allows for adding innovative features without impacting the transactional subsystem and makes it easy the integration of off-the-shelf software. Moreover, it enables the reuse of single subsystems or groups of them in different business contexts of an enterprise. Finally, the pattern suggests a way to exploit the implicit knowledge not only to suggest recommendations to users but also to drive automatic actions (from the Intelligent Mall to the Mall) that aim at satisfying the business objectives of the enterprise.

Particular attention should be given to the proper allocation of business use cases to each subsystem, especially those ones that extend or include use cases scattered across Mall and Intelligent Mall. This integration, which is specific to the application, may be performed at the control layer trying to avoid strong coupling and synchronization. To this end, an event-driven control layer is suggested to capture the events coming from the user interface and to propagate them to the specific subsystems, so preserving a high degree of concurrency and low degree of coupling.

The integration between the two subsystems could be performed by exploiting one or more of the following approaches: (a) Hard Coded integration; (b) Extract, Transform, Load (ETL) techniques; (c) Data Virtualization technologies; (d) Enterprise Service Bus (ESB).

The simplest approach (in term of learning curve) is based on the hard coded implementation of the KE that periodically accesses to the functional interface exposed by the transactional system. Several problems need to be taken into account: timing policies (a short period causes continuous access to the transactional subsystem; long period causes potential inconsistency); the list of functions to be invoked (changes or additions of the features in the Subject would require the revision of the source code of the KE subsystem).

ETL consists of three steps: 1) data extraction from heterogeneous data sources; 2) data transformation for the target; 3) data loading into target. This technique assumes the inconsistency between the data source and the destination be tolerable, but this is not true in modern eCommerce applications since they need providing users with up-to-date information through recommenders.

Data virtualization allows for accessing heterogeneous data systems providing a global integrated view. This technology uses functional interfaces or data access as abstract sources through specific connectors (ODBC, JDBC, REST, SOAP, etc). This integration allows for real-time data accesses but may generate access conflicts with the transactional sub-system. The problem can be alleviated by using caching techniques that reduce the number of concurrent accesses towards the transactional subsystem, keeping the already retrieved data in a cache, so ensuring significant performance gains.

The integration through ESB allows for a strong decoupling and a reuse of single components. This technology provides a messaging system that is able to asynchronously exchange information between publishers

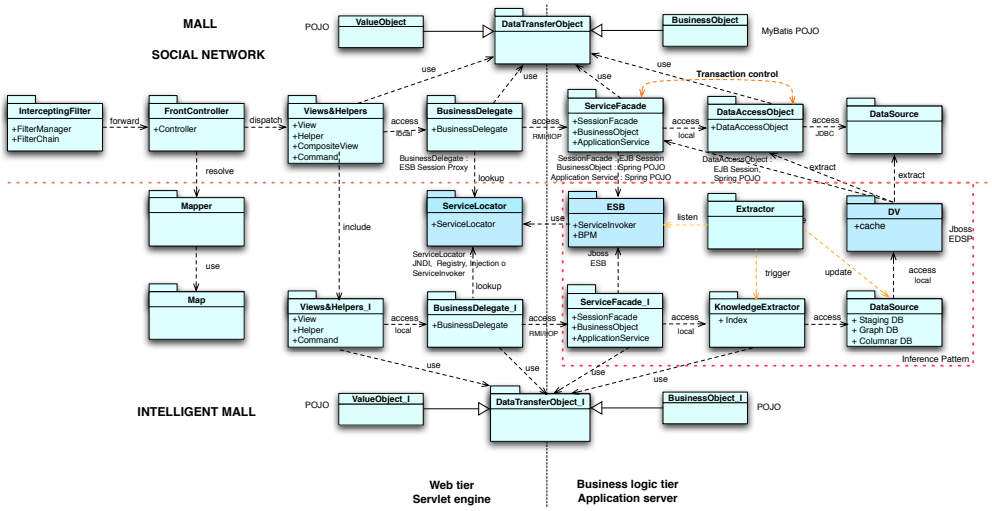

FIG. 4.3. *Integrating the Inference Pattern with the enterprise patterns*

(producers) and subscribers (consumers), so improving concurrency and reducing coupling.

Since ESB is typically not exploited for massive volumes of data, we decided to combine ESB, to communicate events, with (caching-based) data virtualization, for retrieving big data. This solution enables both real-time access to transactional data and loose coupling among the sub-systems, ensuring, at the same time, a strong reuse of services.

Fig. 4.3 shows the integration of the architectural pattern proposed in this paper with the well known enterprise patterns [4] typically exploited to design applications according to multi-tier architecture design.

Following this architecture, software components are classified on the basis of their possible deployment, so identifying two main classes: (a) Web and (b) Business logic components. The application of the Inference Pattern splits the packages in two vertical groups (Mall and Intelligent Mall) and identifies the main packages involved in its implementation (red dotted box).

4.3. Technical architecture and deployment. Following the structure view shown in Fig. 4.3, the implementation of the application has been performed by exploiting specific technology that already offers mechanisms that ease the implementation of the enterprise patterns.

The presentation layer has been developed with GWT (Google Web Toolkit), whereas the business logic has been developed with Spring Framework and MyBatis to interface a MySQL Database. Almost all of the functional components are invoked by Enterprise Java Beans. The data Layer has been built upon an RDBMS (MySQL server), whereas a non-relational persistence system (Neo4J Graph DBMS) has been exploited to improve the performance of the Social Analyzer hosted by the Knowledge Extractor.

Fig. 4.5 shows the technical view of the InViMall architecture. This view points out the logical subsystems (organized in layers) and the kinds of servers and frameworks used to develop the platform. The presentation logic exposes the business functions through HTTP, exploited by both browsers and apps.

The data of the Mall subsystem are shared with the Intelligent Mall subsystem through a staging database, implemented by using the data virtualization mechanisms offered by JBoss Enterprise Data Services (Teiid). It allows for accessing heterogeneous data systems providing a global integrated view.

The implementation of data virtualization has been built on three levels (see Fig. 4.4). The bottom level represents the physical layer that contains the databases of Mall Manager, Social Network and the web services exposed from the legacy eCommerce subsystem adopted (Magento). The middle level abstracts the data sources with one to one mapping from DB to web services data. At this level, a data type transformation function is used to improve the accuracy of data type definition. The top level is composed of a view of the different data sources which are incorporated in a global view to represent the virtual database. This is used only for

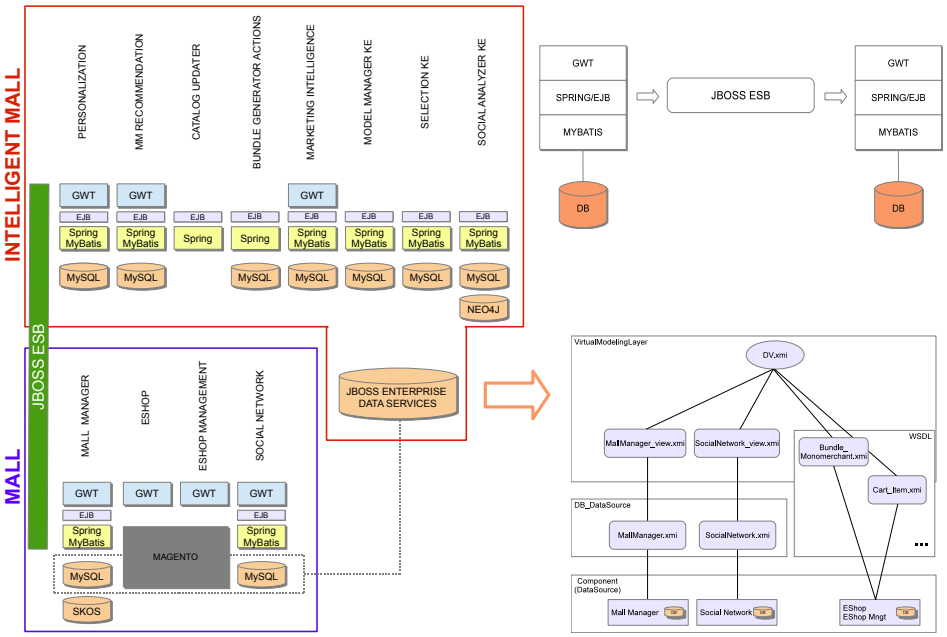

FIG. 4.4. *Technological view: components, data virtualization configuration and communication*

retrieving data while any insert/update/delete operation is performed through the ESB services that wrap the mall functions.

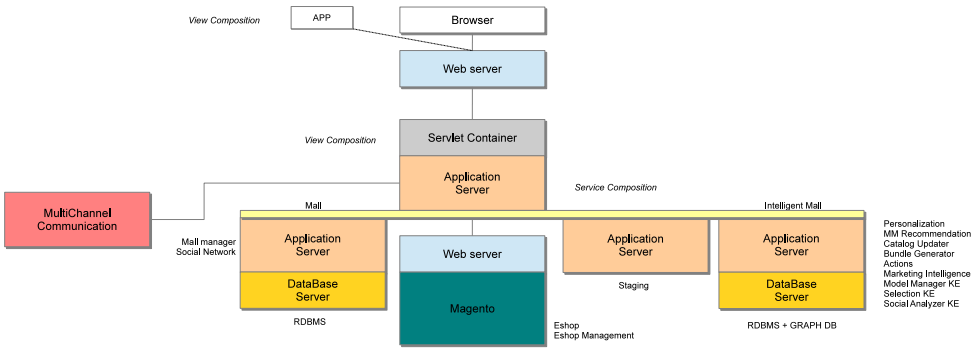

FIG. 4.5. *Technical view: hosting environments*

The components of the InViMall system expose a stateless Session Bean for each interface, using a DAO Layer (Data Access Object) based on MyBatis 3.1.1 for the access to the database. The communication between the components is performed through the JBoss ESB (see Fig. 4.5 (b)).

The whole system has been deployed on virtual machines that are placed on four different virtual LANs. The technological stack used allowed for generating several software deployment packages: a .war artifact for the integrated web tier; an .ear artifact for each functional subsystem implemented through EJB-Spring-MyBatis components; a .vdb artifact for the virtual database; an .esb artifact for packaging the services connected with the ESB. In particular, according to the Inference Pattern, the developed artifacts were deployed onto the virtualized system architecture shown in Fig. 4.6 and organized as follows:

- vLAN D0 (Static presentation tier) hosts an Apache web server for HTTP dispatching and static HTML

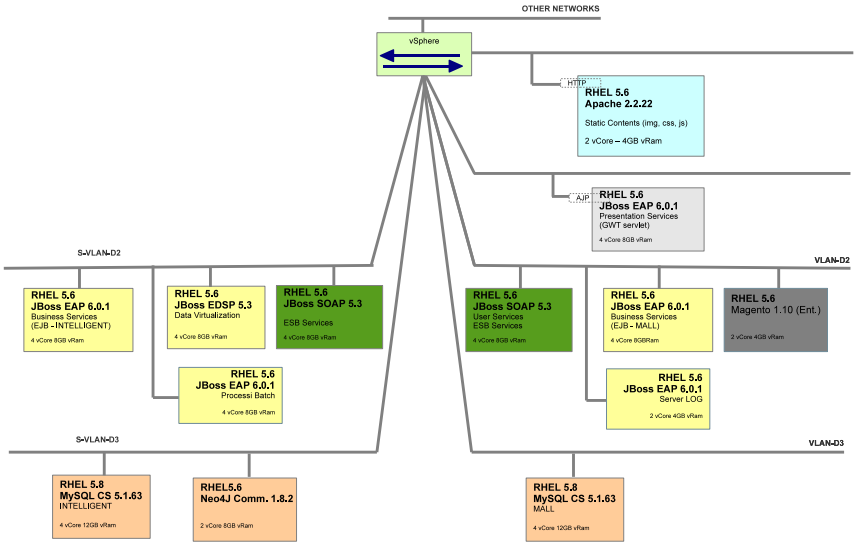

FIG. 4.6. *Application deployment*

pages handling;

- vLAN D1 (Dynamic presentation tier) hosts an Application Server for the integrated GWT Servlets;
- vLAN D2 (Business logic tier) is switched in two segments and hosts: on the right side, an application server for the ESB, an application server that runs the Mall business logic, a log server, and Magento; on the left side, an application server for the ESB, an application server that runs the Intelligent Mall business logic, an application server to host the data virtualization layer, and an application server that runs the batch processes;
- vLAN D3 (Data management tier) is also switched in two segments and hosts: on the right side, a MySQL DBMS for storing the Mall data; on the left side, a Neo4J Graph-based DBMS for storing social data, and a MySQL DBMS for storing data for recommendation and decision support extracted from the virtualized data base or from the Graph-based DB through batch processing.

It is worth to note, that the deployment analyzed before is a simplification of the real deployment, which considers also the replication of each tier to address reliability and horizontal scalability through replication.

5. Conclusion. The paper presented an architectural pattern that helps software architects to design intelligent enterprise systems that combine interactive features with batch processing to infer knowledge from large amounts of data.

The pattern was applied with success to design a complex application that implements an electronic mall. Thanks to it, software architects clearly and rapidly identified the functional subsystems, by associating to them use cases. It, also, contributed to create a loosely coupled system with a clear logical separation between the transactional and intelligent sub-systems, and a further separation among the subsystems of the intelligent side.

The benefits of the pattern were discussed also with respect to the integration techniques that can be exploited to efficiently implement the pattern. The possibility to deploy the subsystems onto different resources with specialized characteristics makes the overall system efficient and flexible, since additional subjects can be easily added. Scalability is achieved by properly deploying transactional and batch oriented subsystems onto different hardware resources linked to switched network segments.

The pattern enables the integration of both real-time, event-driven interactions, and data accesses for large volume of data. In the future, the application of more sophisticated integration middleware platforms that transparently combine the two benefits above will be investigated [23].

Acknowledgements. This work was partially supported by Italian Ministry of Economic Development in the context of the Intelligent Virtual Mall (InViMall) Project MI01-00123.

REFERENCES

[1] http://www.opengroup.org/togaf/
[2] http://www.ibm.com/developerworks/patterns/
[3] http://www.opengroup.org/togaf/
[4] http://martinfowler.com/articles/enterprisePatterns.html
[5] Hohpe G., Woolf B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison Wesley (2003)
[6] Gamma E., Helm R., Johnson R., Vlissides J.: Design Patterns - Elements of Reusable Object-Oriented Software, Addison-Wesley (1995)
[7] Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M.: Pattern-Oriented Software Architecture - A System of Patterns, John Wiley & Sons, (1996)
[8] Potel M., MVP: Model-View-Presenter: The Taligent Programming Model for C++ and Java. (1996)
[9] Plakalovic D., Simic D.: Applying MVC and PAC patterns in mobile applications. Journal of Computing, 2(1) (2012)
[10] Fowler M. J.: Patterns of Enterprise Application Architecture. Addison Wesley (2002)
[11] Cong S., Hunt E., Dittrich K. R.: IEIP: An Inter-Enterprise Integration Platform for e-Commerce Based on Web Service Mediation. In: European Conference on Web Services - ECOWS, pp. 201-210 (2006)
[12] Kephart J., Chess D.: The vision of autonomic computing. Computer, 36(1):41-50, (2003)
[13] Huebscher M. C, McCann J.A.: A survey of autonomic computing - degrees, models, and applications. ACM Computing Survey, 40(3):1-28 (2008)
[14] Ansari S., Kohavi R., Mason L., Zheng Z.: Integrating E-Commerce and DataMining: Architecture and Challenges. In: International Conference on Data Mining, IEEE, pp. 27-34 (2001)
[15] Birtolo C., Ronca D., Armenise R.: Improving accuracy of recommendation system by means of Item-based Fuzzy Clustering Collaborative Filtering. In: 11th International Conference on Intelligent Systems Design and Applications - ISDA '11, IEEE, Spain (2011)
[16] Birtolo C., De Chiara D., Ascione M., Armenise R.: A Generative Approach to Product Bundling in the e-Commerce domain. In: the 3rd World Congress on Nature and Biologically Inspired Computing - NaBIC'11, pp. 169-175. IEEE, Spain (2011)
[17] Birtolo C., Diessa V., De Chiara D., Ritrovato L., Customer Churn Detection System: Identifying customers who wish to leave a merchant. In: 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, LNCS Springer, Amsterdam, June 17-21, pages: 411-420 (2013)
[18] Bachmann F., Bass L.: Introduction to the Attribute Driven Design Method. In: the 23rd International Conference on Software Engineering - ICSE (2001)
[19] Clements P., et al.: Documenting Software Architectures: Views and Beyond. 2nd edition, Paerson Education (2011)
[20] Yu H., et al.: Knowledge Management in E-commerce: A Data Mining Perspective. In: Management of e-Commerce and e-Management, IEEE (2009)
[21] Guo S., Wang M., Leskovec J.: The Role of Social Networks in Online Shopping: Infor-mation Passing, Price of Trust, and Consumer Choice. In: Conference on Electronic Commerce, ACM (2011)
[22] Polese M., Tretola G., Zimeo E.: Self-adaptive management of Web processes. In: Web Systems Evolution (WSE), IEEE, pp. 33-42 (2010)
[23] Zagarese Q., Canfora G., Zimeo E., Alshabani I., Pellegrino L., Baude F., Efficient data-intensive event-driven interaction in SOA. In: Symposium on Applied Computing - SAC 2013, ACM, 1907-1912, (2013)

Edited by: Viorel Negru and Daniela Zaharie
Received: June 2, 2013
Accepted: Jul 9, 2013

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.