jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2015, VOL. 61, NO. 1, PP. 17–23
Manuscript received February 15, 2015; revised March 2015. DOI: 10.1515/eletel-2015-0002

A Software Architecture Assisting Workflow Executions on Cloud Resources

Grzegorz Borowik, Marcin Woźniak, Andrea Fornaia, Rosario Giunta, Christian Napoli, Giuseppe Pappalardo, and Emiliano Tramontana

***Abstract*—An enterprise providing services handled by means of workflows needs to monitor and control their execution, gather usage data, determine priorities, and properly use computing cloud-related resources. This paper proposes a software architecture that connects unaware services to components handling workflow monitoring and management concerns. Moreover, the provided components enhance dependability of services while letting developers focus only on the business logic.**

***Keywords*—software architecture, dependability, workflows, cloud computing, monitoring**

I. Introduction

NOWADAYS, business enterprises organise their provided operations by means of a pre-defined *workflow*, i.e. a flow of execution relating activities supported by software services and regulated by an engine [1], [2]. Generally, several services connect with others, or provide data to others, according to a predefined workflow, while sharing common computing resources, available e.g. as a cloud-computing facility [3], [4]. By resorting to cloud-computing, transparent access e.g. to shared services, hardware and data is made possible, thus enabling a higher level of *availability* as well as higher performances [5].

One of the primary needs for enterprises is to handle service executions in such a way to: (i) monitor human activities, (ii) have a smooth execution on servers, and (iii) achieve a defined degree of *dependability* [6]. For the latter, when a server handles numerous requests, to counteract slowing responsiveness and enhance *availability*, services can be activated on other cloud resources [7] or by means of software agents [8]. For an enterprise, *monitoring* their workflows means gaining metrics related to business service performances, employee productivity, etc. Such metrics enable decision making for optimising how human resources provide their assistance, handling priorities, such as the allocations of tasks to human resources or processes to hosts.

Services activated by means of a workflow could include the additional code that supports both dependability and monitoring issues, however such a solution would bring a high complexity level for services, lower *maintainability* and increase development costs [9]–[11]. Moreover, when using other traditional solutions for supporting additional concerns, such as non-functional ones, services have to conform to an ad-hoc supporting framework or development model [12], [13]. This limits *modularity* and forces some components to be manually adapted. There are also some efficient ways to improved state encoding for structures with embedded memory blocks [14]. In order to support modularity, we propose a software infrastructure that seamlessly activates workflow execution, provides services with *monitoring* and enhances *dependability*. The technology empowering the said seamless integration is *aspect-orientation* (AO), defined in [15], [16]. A software *aspect* is a module that includes portions of code (comparable to methods) that are injected into an existing component according to defined rules. AO systems have been built to separate QoS-enhancing code from functional code, design pattern-related code from classes [17]–[19] etc. Unlike previous approaches (see Section V for a detailed comparison) our proposal can be applied to services, and workflows, without relying on any assumptions, is not intrusive for the underlying support (such as JVM or OS libraries), enhances modularity, and is not disrupting in terms of execution flow, development process and freedom for service developers.

The paper is organised as follows. Section II provides the motivation and an overview of our approach. Section III describes the proposed software components that automatically provide workflow management, and enhance services with monitoring and dependability. Section IV introduces our strategy for the use of cloud resources. Section V discusses the related work, and Section VI draws our conclusions.

II. Proposed Approach

A. Analysed Enterprise Scenario

A typical enterprise workflow is generally formalised by means of a description, which is given to a workflow engine, e.g. JBOSS jBPM[1], that timely starts the several services described [1]. The workflow engine is able to send notifications about the state (e.g. just started, executing, finished) of the services on a workflow, and therefore alert humans or gather statistical data about the execution.

Figure 2 shows an example of a simplified workflow, dubbed city planning, whereby several services are executed each corresponding to a step that can be performed after

This work has been supported by project PRISMA PON04a2 A/F funded by the Italian Ministry of University within PON 2007-2013 framework.

G. Borowik is with Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland (e-mail: gborowik@tele.pw.edu.pl).

M. Woźniak is with Institute of Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland (e-mail: marcin.wozniak@polsl.pl).

A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, and E. Tramontana are with Department of Mathematics and Informatics, University of Catania, Viale A. Doria 6, 95125 Catania, Italy (e-mails: fornaia@dmi.unict.it, giunta@dmi.unict.it, napoli@dmi.unict.it, pappalardo@dmi.unict.it, tramontana@dmi.unict.it).

[1]http://www.jboss.org/jbpm

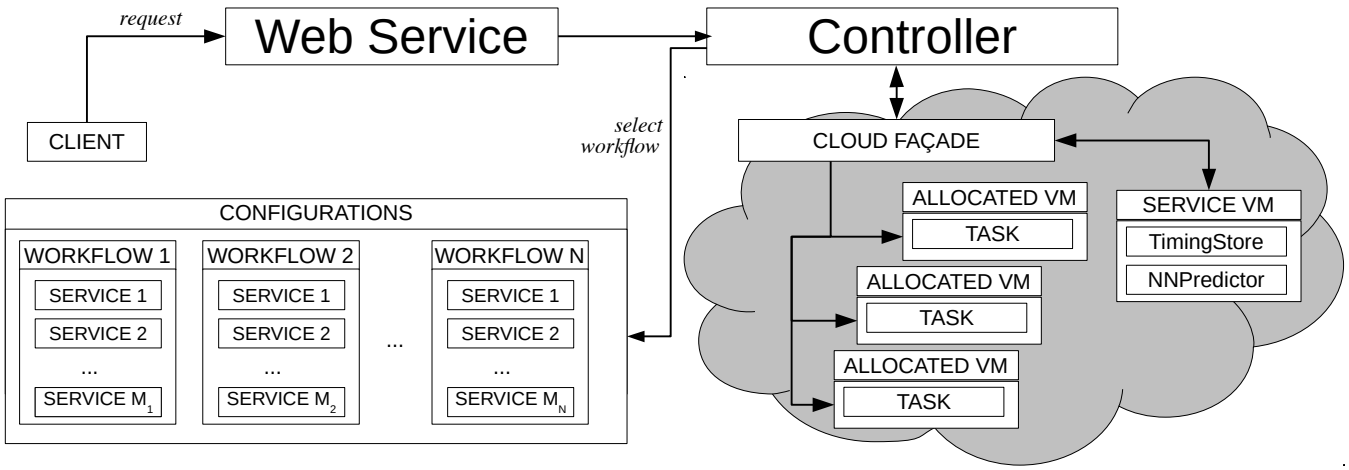

Fig. 1. The overall configuration of the developed system

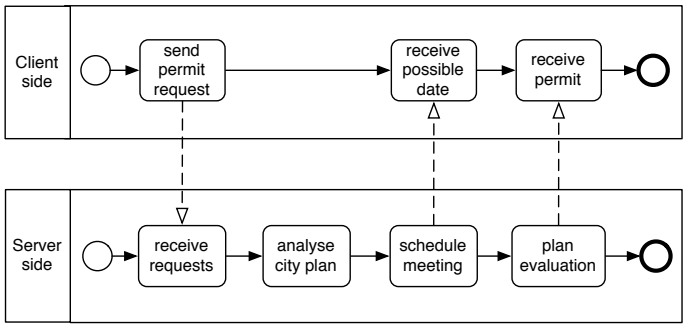

Fig. 2. An example of workflow city planning on a distributed system

receiving a user request for a permit. A reference model for the software system assisting such steps will have one or more client applications enabling the user to submit a request or to gather replies. Hence, e.g. the step send permit request could be performed using a web browser or an ad-hoc application connecting to a service, receive possible date could be a message received as a status update on a web page or an email, etc. Services on the server side are processes running, or started, according to the indication given by the workflow steps, hence e.g. receive requests is the first step of an ad-hoc workflow, and is a process listening for incoming requests, analyse city plan is a process started as a second step of the workflow once the previous step has been performed, etc.

Services within a workflow are generally of different nature, e.g. have their own data or processing requirements, hence they should be handled differently when dealing with dependability. Let us suppose that analyse city plan is a resource consuming process whose execution time has to be guaranteed, instead, another service could simply provide immutable stored documents. Then, handling requests that trigger service execution requires the provisioning of ad-hoc computing resources to ensure dependability.

B. Transparently Provided Support

Cloud-based solutions provide a means to limit costs for an enterprise needing high-performance hardware resources [3]. Therefore, the server-side software components can be partially or wholly supported by cloud-based resources. In such a scenario, additional concerns can be identified, as *workflow monitoring* and *dependability enhancements*. Our contribution provides such features by inserting support into existing services.

Workflow monitoring. Business related services are automatically provided with monitoring, i.e. how and when such services are used. Monitoring allows typical usage behaviour of enterprise employees to be detected, thanks to their interactions with services. This makes it possible to compute metrics on productivity, workflow execution time, and potential bottlenecks of workflows. Data gathered while monitoring services and workflows are sent to components located on cloud-based resources, such that even though large amount of data are accumulated, they can be easily processed. Therefore, our monitoring concern is different from cloud metering services that observe resource consumption and exhibit the status of several resource-based metrics. It is also important to use some efficient methods to accumlate input data. Fast aggregation methods for large data sets are presented in [20] and [21].

Enhancing dependability. As a means to improve *availability*, we resort to an automatic and transparent support that detects whether a service to be executed needs to be assisted by secondary, cloud-based hosts, which execute services and provide contents. By delegating a portion of the service processing requested to the secondary host, we enhance overall response times and limit the workload on hosts. The further benefit given by the use of secondary hosts is a marked *reliability* increase for the whole system, for additional hosts can easily take over the workload assigned to one that incidentally incurs into faults. This makes the overall enhanced system capable of providing uninterrupted service under load peaks and faults.

III. Software Infrastructure

In order to support the features outlined above, we propose a few software components that enhance services and provide: (i) monitoring of workflow services, (ii) resource management, (iii) dependability.

The connection between provided components and workflow services is accomplished by means of *aspect-oriented* programming. A software *aspect* is a component defining *pointcuts* and *advices* [15], [16]. The pointcuts trigger execution of advices when given join points, i.e. some points on the code of a service, are executed. Aspects allow *crosscutting concerns* to be dealt with in a modular way, thus making components maintainable and prone to be reused [18], [19], [22].

Figure 3 shows how our components operate, as seen by an external observer. The chief actors are: the deployed services, such as Web-Service1 (the middle layer), which clients can connect to; a set of cloud-servers, such as Cloud-Server1 (the rightmost layer); a set of regular clients. Services, comprising their business logic, can be exposed on the web, hence users can interact with their browser, or reside on a cloud resource. Our provided aspect Reporter connects services exposed on the web with other cloud-based services. For this, an ensemble of several supporting classes are used, i.e. mainly ReqHandler, Scheduler, NNPredictor, LocalTimingStore.

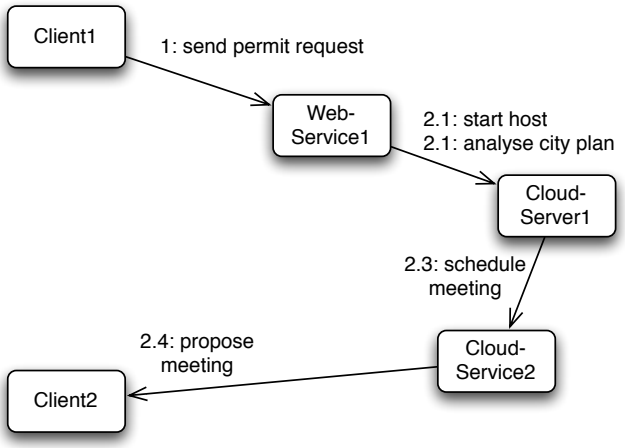

Fig. 3. Client requests served by a web server and a cloud-server

A. Workflow Descriptor

In the example above (see Figure 2), the steps of a defined workflow are associated with indications that allow resource allocations. Table I provides the workflow services with such additional data.

In our solution, a component, dubbed Controller, takes as input the above data and accordingly during runtime handles the current request. Typically in our scenario, a new instance of a workflow is started by means of a request coming from a web service (see the following section), then Controller is alerted, finds the workflow that has to be executed, and starts operations on cloud resources for the following services.

TABLE I
The Characterisation of Services for Workflow City Planning in Order to Manage of Cloud Resources

service name	start, end status	resource type
receive requests	on, on	no cloud
analyse city plan	off, off	dedicated, large VM
schedule meeting	standby, standby	shared, small VM
plan evaluation	off, off	shared, medium VM

In the above example, after executing service receive requests on a web server, Controller receives an alert, finds the related workflow and prepares resources for the execution of service analyse city plan by starting a dedicated VM. From the second row in the above table, we can see that the service has to be started on a dedicated large VM.

For each workflow, additional data have to be provided, such as e.g. the priority, the allowed deadline, the number of instances that can be concurrently executed. Moreover, for each service on the workflow, the number of its instances that can be concurrently active is also given.

B. Aspect Reporter

Aspect Reporter enhances a service responding on a known IP address, and transparently provides it with abilities to effectively handle request bursts. Such an aspect relies on other components that are located on cloud-based servers. Aspect Reporter connects with services listening HTTP requests, monitors the timing of their execution, and triggers execution of other offloading services when needed. Once Reporter has gathered workflow-related metrics, data are sent to the local class LocalTimingStore. In turn, such data will be periodically sent to service TimingStore located on a cloud server. The latter appropriately merges data coming from different users and services, or different instances of the same service. Figure 4 shows an UML sequence diagram for the said measuring operations, since when Reporter pointcut trapRequest() intervenes into the execution of a service.

We have defined pointcuts for capturing a variety of possible implementations, because an HTTP request can be handled by a servlet, or as a EJB. Servlet implementations, which handle HTTP requests, are subclasses of HttpServlet class (available into Java package javax.servlet.http) and have to have methods doPost() or doGet(). Whereas when having services implemented as EJBs, annotation @WebService is used, or to expose only a method as a part of a web service, annotation @WebMethod is used (such annotations are defined into package javax.jws). The corresponding *pointcuts* trapRequest() and trapRequestWebServ() are shown in the listing of Figure 5. The first pointcut captures all the points of the program that invoke methods doPost() or doGet() implemented in a subclass of HttpServlet; whereas the second pointcut captures all the invocations to any method of a class that has been marked with annotation @WebService.

The timing of operations exposed as a web service gives a great amount of details on the workflow activities that are performed. This aspect encapsulates a concern that is crosscutting both for the components of a service, and for several services available within workflows defined by the enterprise.

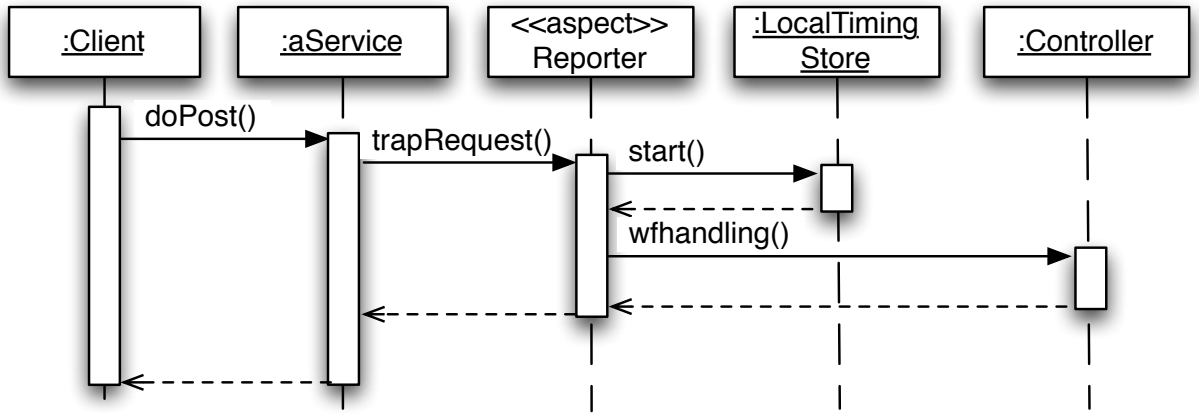

Fig. 4. Sequence diagram showing aspect Reporter that monitors invocations to a service performed by a client class Client and connect the proper workflow related service

```
public aspect Reporter {
    private final LocalTimingStore lts =
                    LocalTimingStore.getInstance();

    // capture method invocations on a servlet
    pointcut trapRequest() :
                    call(void HttpServlet+.doPost(..)) ||
                    call(void HttpServlet+.doGet(..));

    void around() : trapRequest() {
        HttpServlet t = (HttpServlet) thisJoinPoint.getTarget();
        Object[] arg = thisJoinPoint.getArgs();
        HttpServletRequest r = (HttpServletRequest) arg[0];
        lts.start(t, r, System.nanoTime());
        if (!Controller.getInstance().wfhandling(t))
                        proceed();
        lts.end(t, r, System.nanoTime());
    }

    // capture method invocations on annotated classes
    pointcut trapRequestWebServ() :
                    call(@WebService * *.*(..));

    void around() : trapRequestWebServ() {
        Object t = thisJoinPoint.getTarget();
        lts.start(t, System.nanoTime());
        if (!Controller.getInstance().wfhandling(t))
                        proceed();
        lts.end(t, System.nanoTime());
    }
}
```

Fig. 5. Aspect Reporter that intercepts invocations on operations of classes implementing a service exposed as HTTP and records the starting and finishing time

Nevertheless, the code of such an aspect is independent of any observed services. Compared with alerts that can be set up on the workflow engine to be notified of service execution, aspect Reporter additionally gives a means to control operations within a service, hence supports fine-grained monitoring and adaptivity. Thanks to this aspect, workflows and services can be given resources according to different policies.

C. Dependability and Flow of Operations

For the proposed solution to be effectively used, aspect Reporter needs to be deployed enterprise-wide. I.e. aspect Reporter has to be deployed on every service to be enhanced with the monitoring and dependability support. Aspect Reporter temporarily stops an incoming HTTP request and Controller determines whether the request should be handled by the local host, or by another service on a cloud resource. How to handle a request depends on the current load, request type, requesting user, and the applicable workflow. For each service, Controller holds the configuration needed, including: priority, number of allowed concurrent instances, flavour and status of the VM, etc. (see Section III-A). From such data and the actual status Controller determines whether to let the request forward, start another VM, etc.

Controller is the main component assisting Reporter. Figure 6 shows the relevant interactions, i.e. when a request arrives on a handling web service, Reporter aspect intervenes to let the request be served according to our model. Hence, pointcut trapRequest() is activated by rules defined as in Figure 4 and the listing in Figure 5. Such an aspect then extracts data to identify the request. The activated advice checks with Controller the priority and whether the current request is allowed and can continue execution on the local host (see call to Controller.wfhandling()). The check is handled by first calling getResources() (Figure 6), and this finds which service and host are needed to serve the response.

When the host on the local web server can not handle the request, then the list of needed resources is passed to CloudFacade class, calling its startVM() method or a proper method (such as e.g. enqueRequest(), etc.), which allow the requested service to be later executed on the cloud resource. Class CloudFacade handles the details for the operation to be started on a cloud facility.

Whenever a new request arrives, an hybrid neural network engine [23] is updated for the benefit of future forecasts [2], using the updSeries() call. Moreover, NNPredictor subsystem periodically computes the load estimate as an amount of received requests to services of the enterprise and will boot up or shut down cloud-hosted servers. NNPredictor can estimate up

to six hours ahead with a relative error of 0.6 per thousands [2]. When a subsequent service on the same workflow instance has to be executed, the needed cloud resorces have been started and made ready to serve by NNPredictor. Thus, Controller collects the addresses of which host will be assisting the current session in order to let the following service execute.

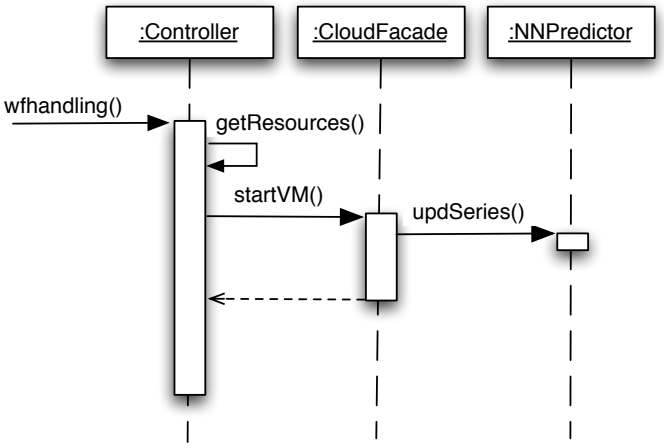

Fig. 6. Sequence diagram showing interactions with a cloud and the resource usage predictor

Each client is identified by a set of characteristics, such as IP address, user id, priority, etc. and may pertain to different categories, such as response time, connections per second, etc. Accordingly, Controller determines whether to accept the current request, and in case it is allowed compute a priority for all the services of the related workflow. The priority is based on the identity of the requester and the workflow whereby the current request belongs, and in turn according to such a priority cloud resources (e.g. VM flavours) are selected.

As far as the load balancing part is concerned two important benefits are achieved: (i) the load of a request in a workflow instance is distributed according to the several hosts involved in the execution, and their state, (ii) initial requests can be accepted, put on hold or rejected, by our Controller.

The following section describes our strategy to allocate resources on a cloud when multiple requests have to be handled, and in order to minimise the number of allocated resources, thus effectively using in-house facilities, without resorting to additional costs.

IV. Policy for Allocating Resources

Quantum electrodynamics inspired us an ad-hoc description for a computational cloud which we have used as the basis to model an evolutionary algorithm [24]–[26] in order to follow the dynamical state of the system during time. If we define a computational cost for a task (such a cost could refer to memory, cpu time, complexity or other software measurements), then it is possible to use the defined cost for an appropriate fitness function. Since we are interested in finding the optimum solution for the allocation of resources on a cloud, it would be equivalent to compute the local minima of such a fitness function, once taken into account all the constraints of the case. Such a problem is similar to a typical partitioning problem for a fermion particle gas, therefore solvable with a few basic tools from quantum electrodynamics.

In our description, then, the said computational cost will have the same meaning of the energy of a fermion. Therefore, the resources provided by a VM can be seen as the local minima of a potential cost field. Similarly to fermions that naturally tend to the minimum energy state, then our cloud must evolve in order to reach the minimum cost state.

In particle physics the fermions exhibit antisymmetric wave functions, beside the mathematical formulation and the physical meaning, front this fact depends what is generally known as the Pauli exclusion principle. Such a principle states that in a quantum system two fermions can not share the same pure state, in facts the Pauli exclusion principle is a selection rule for forbidden states. While in quantum mechanics such a rule is taken into account as a brute fact, this natural feature of our universe can as well give us the solution to the considered problem.

Within the parallelism among cloud computing and quantum electrodynamics, we will describe the available resources of a cloud as points of local minima for a fitness function. We decided then to use as a fitness function the total energy of a quantum system defined by a partition function for fermion particles, therefore describing the system by a Fermi-Dirac statistical distribution.

We ordered the resources according to their related potential cost, so that the resource R_i covers a less or equal potential cost with respect to R_{i+1}. Then we have imagined the resource requests as free particles, while we described an assigned resource as a bound unoccupied state. Therefore, inspired by the statistical thermodynamics of a fermions gas, we defined an occupation function.

Let us suppose to have N_r resources $R_i \in R$ waiting to be occupied in order to fulfil N_q requests $q_j \in Q$. The sets R and Q are continuously populated each time new resources are freed (or added) to the cloud and each time new resource requests are made. Suppose also that while time goes pairs (r_i, q_j) are formed. In this case, if we define $\epsilon(r_i)$ the potential cost of a resource r_i, and $\epsilon(q_j)$ the effective cost[2] of a request, then the occupation function will be algorithmically populated by using the following rule:

$$Z_n = (r_i, q_j) : \epsilon(r_i) \geq \epsilon(q_j) \quad \forall\ Z_n \in R \times Q \tag{1}$$

where Z_n is the n-esime occupation. For each Z_n then we can obtain an energy difference defined as $\delta : R \times Q \to \mathbb{R}$ so that

$$\delta(Z_n) = \delta(r_i, q_j) = \epsilon(q_j) - \epsilon(r_i) \quad \forall\ Z_n \in Q \times R \tag{2}$$

We then define a fitness function $f(\tau) : R \times Q \to \mathbb{R}$ where τ identifies a discrete time step, so that

$$\left[f(Z_n)\right]_\tau = \left[f(r_i, q_j)\right]_\tau = \left[\delta(r_i, q_j) - w_i\right]_\tau \tag{3}$$

with $[w_i]_0 = 0 \quad \forall\ r_i \in R$ if $\tau = 0$ is the first time step. By means of this fitness function, it is possible to identify a subset of perfectly matching pairs $\Omega = \{Z_n : [f(Z_n)]_0 = 0\}$. Moreover, we can define the set of non matching pairs as the

[2]We intend as affective cost the real cost that will be paid when chosing resource r_i.

complementary set $\Theta = \{Z_n : Z_n \notin \Omega\}$. While Ω identified the pairs of resources and requests that naturally permit us to waste virtually no resources, it is still needed to find the minimum configuration for the pairs in Θ.

Therefore, at each time step a gradient descent algorithm [27]–[29] is applied in order to modify the coefficients $[w_i]_\tau$ so that:

$$[w_i]_{\tau+1} = \left[w_i - \eta f(Z_n) \frac{\partial f}{\partial w} \right]_\tau \quad \forall\ Z_n \in \Theta \tag{4}$$

where η is a fixed step size.

Coming back to the imagined fermion gas, this gradient descent algorithm is equivalent to the natural motion caused by the difference of energy with respect to the bound state which is therefore unstable. By imposing as constraint (as it is in nature) that all the free particle states have higher energy with respect to the bound states, then the system naturally evolves to the lowest energy overall state. The same concept is applicable then to the potential cost corrected with the terms $[w_i]_\tau$.

It comes trivially that the system also evolves naturally to the optimum, moreover adding or removing resources and requests to the system does not modify the dynamic evolution of the system, but, in this latter case, before to apply the evolutionary algorithm expressed by (4), it could be necessary to recompute the population of Ω and Θ.

Algorithm 1 Minimization Gradient Descent Algorithm

```
Start,
Define fitness condition f(·),
Define step size η and time horizon T,
Crossmatch the resources and the requests,
Compute the perfectly matching pairs set Ω,
Compute the complementary set Θ,
t = 1,
while t ≤ T do
    Compute [w_i]_t using (4),
    t++
end while
Return Ω ∪ Θ,
Stop.
```

The adopted gradient descent algorithm is a local optimization technique generally based on algorithmic attempts. Such techniques permits to obtain the optimum configuration with a very simple procedure, therefore computationally advantageous with respect to more computationally expensive analytical calculations.

The presented numerical algorithm searches for a solution in the defined space by means of recurrent adjustments dependent by the gradient of the fitness function. In Algorithm 1 the developed gradient descent algorithm is presented. The algorithm starts with the population of the perfectly matching pairs set Ω and by computing its complementary set Θ. In the successive steps it computes the gradient ∇f and then the results of equation (4). It has to be noticed that it is the gradient sign that determines the direction of the function descent. At the end of each time step the new configuration of Ω is computed, and the algorithm is iterated until a time horizon T is reached, or it comply with a predefined stop condition.

V. Related Works

A thorough survey of existing literature on dependability, QoS and performance for internet applications is reported in [30].

The work in [31] proposes the instrumentation of the underlying system socket library, as a means to intercept inbound and outbound traffic, so as to monitor service and performance parameters. Served content is adaptively degraded to comply with SLAs. The approach mixes communication and QoS concerns.

An other way to improve QoS is to use more efficient computing units, that can provide optimal energy characteristic of a processor allocator and a Network-on-Chip [32] or tailored finite state machines with programmable structures [33].

The development cycle described in [13] starts from a set of (suitably modelled) QoS constraints and, after several model-transformations, produces the final system, consisting of separate aspects implementing exclusively the QoS concern. A similar approach is reported in [34]. It is also based on a non-standard, design-level modeling language and produces a final application enjoying an advanced modularity degree. While such approaches are useful to produce a modular system, they lack the capability of enhancing existing components, as instead our approach allows, thus ruling out their utilisation in business contexts where re-development costs have to be kept at minimum and reuse is the norm. Therefore to have an optimal QoS we often use tailored positioning based on methematical modlling. A detailed mathematical modelling for positioning queueing systems possible to widely apply in cloud computing is given in [35], [36] and [37].

VI. Conclusions

In this paper we proposed an architecture that tackles the need of starting workflow services, while monitoring their execution and controlling the used resources. Services that are part of a workflow can be controlled by our proposed aspect-oriented solution and make these able to automatically resort to cloud resources, without any reengineering effort. Aspects can collect timing related data about the usage of any service involved in the workflow, thus allowing the computation of several metrics about the usage of the services. The automatic scaling on cloud resources, whose number is estimated by a neural network predictor, is performed by distributing requests according to a fitness function that minimises costs. Thus enhancing the availability of services (appropriately replicated on cloud hosts) without having to bear the cost due to in-house hosting.

References

[1] T. Erl, *SOA design patterns*. Pearson Education, 2008.
[2] C. Napoli, G. Pappalardo, and E. Tramontana, "A hybrid neuro-wavelet predictor for qos control and stability," in *Proceedings of AIxIA*, ser. LNCS, vol. 8249. Springer, December 2013, pp. 527–538.

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[4] R. Tolosana-Calasanz, J. Á. BañAres, C. Pham, and O. F. Rana, "Enforcing qos in scientific workflow systems enacted over cloud infrastructures," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1300–1315, 2012.

[5] C. Napoli, G. Pappalardo, E. Tramontana, and G. Zappalà, "A cloud-distributed gpu architecture for pattern identification in segmented detectors big-data surveys," *The Computer Journal*, p. bxu147, 2014.

[6] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[7] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*. IEEE, 2014, pp. 1077–1084.

[8] C. Napoli, G. Pappalardo, and E. Tramontana, "An agent-driven semantical identifier using radial basis neural networks and reinforcement learning," in *Proceedings of the XV Workshop Dagli Oggetti agli Agenti*, vol. 1260. CEUR-WS, 2014. [Online]. Available: http://ceur-ws.org/Vol-1260/

[9] G. Pappalardo and E. Tramontana, "Suggesting extract class refactoring opportunities by measuring strength of method interactions," in *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*. IEEE, December 2013, pp. 105–110.

[10] E. Tramontana, "Automatically characterising components with concerns and reducing tangling," in *Proceedings of QUORS workshop at Compsac*. IEEE, 2013, pp. 499–504.

[11] C. Napoli, G. Pappalardo, and E. Tramontana, "Using modularity metrics to assist move method refactoring of large systems," in *Complex, Intelligent, and Software Intensive Systems (CISIS), 2013 Seventh International Conference on*. IEEE, 2013, pp. 529–534.

[12] J. Loyall, D. Bakken, R. Schantz, J. Zinky, D. Karr, R. Vanegas, and K. R. Anderson, "Qos aspect languages and their runtime integration," in *Lecture Notes in Computer Science – LCR Workshop*, vol. 1511. Springer, 1998, pp. 303–318.

[13] G. Ortiz and B. Bordbar, "Aspect-oriented quality of service for web services: A model-driven approach," in *Proceedings of ICWS*. IEEE, 2009, pp. 559–566.

[14] G. Borowik, "Improved state encoding for FSM implementation in FPGA structures with embedded memory blocks," *International Journal of Electronics and Telecommunications*, vol. 54, no. 2, pp. 9–28, 2008.

[15] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin, "Aspect-oriented programming," in *Lecture Notes in Computer Science – ECOOP*, ser. LNCS, vol. 1241. Springer, 1997, pp. 220–242.

[16] R. Laddad, *AspectJ in Action*. Grennwich, Conn.: Manning Publications Co., 2003.

[17] F. Banno, D. Marletta, G. Pappalardo, and E. Tramontana, "Tackling consistency issues for runtime updating distributed systems," in *Proceedings of International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE, 2010, pp. 1–8.

[18] R. Giunta, G. Pappalardo, and E. Tramontana, "Handling replica management concerns by means of aspects," in *Proceedings of WETICE*. IEEE, 2007, pp. 284–289.

[19] R. Giunta, G. Pappalardo, and E. Tramontana, "Superimposing roles for design patterns into application classes by means of aspects," in *Proceedings of the ACM Symposium on Applied Computing, SAC*. ACM, March 2012. DOI: 10.1145/2245276.2232082, pp. 1866–1868.

[20] M. Woźniak, Z. Marszałek, M. Gabryel, and R. K. Nowicki, "Modified merge sort algorithm for large scale data sets," *Lecture Notes in Artificial Intelligence – ICAISC'2013*, vol. 7895, PART II, pp. 612–622, 2013.

[21] M. Woźniak, Z. Marszałek, M. Gabryel, and R. K. Nowicki, "On quick sort algorithm performance for large data sets," in *Looking into the Future of Creativity and Decision Support Systems*, A. M. J. Skulimowski, Ed. 7-9 November, Cracow, Poland: Progress & Business Publishers, 2013, pp. 647–656.

[22] R. Giunta, G. Pappalardo, and E. Tramontana, "AODP: refactoring code to provide advanced aspect-oriented modularization of design patterns," in *Proceedings of SAC*. ACM, March 2012, pp. 1243–1250.

[23] G. Capizzi, C. Napoli, and L. Paternò, "An innovative hybrid neuro-wavelet method for reconstruction of missing data in astronomical photometric surveys," in *Artificial Intelligence and Soft Computing*. Springer Berlin Heidelberg, 2012, pp. 21–29.

[24] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszalek, D. Polap, and M. Wozniak, "Simplified firefly algorithm for 2d image key-points search," in *Computational Intelligence for Human-like Intelligence (CIHLI), 2014 IEEE Symposium on*. IEEE, 2014, pp. 118-125.

[25] M. Woźniak, M. Gabryel, R. K. Nowicki, and B. Nowak, "A novel approach to position traffic in nosql database systems by the use of firefly algorithm," in *Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems*, G. A. Papadopoulos, Ed. 6-8 November, Limassol, Cyprus: University of Cyprus Press, 2014, pp. 208–218.

[26] M. Woźniak, "On positioning traffic in nosql database systems by the use of particle swarm algorithm," in *Proceedings of XV Workshop DAGLI OGGETTI AGLI AGENTI – WOA'2014*. 25-26 September, Catania, Italy: CEUR Workshop Proceedings (CEUR-WS.org), RWTH Aachen University, 2014, paper 5.

[27] F. Bonanno, G. Capizzi, and C. Napoli, "Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*. IEEE, 2012, pp. 941–945.

[28] G. Capizzi, F. Bonanno, and C. Napoli, "Recurrent neural network-based control strategy for battery energy storage in generation systems with intermittent renewable energy sources," in *Clean Electrical Power (ICCEP), 2011 International Conference on*. IEEE, 2011, pp. 336–340.

[29] F. Bonanno, G. Capizzi, S. Coco, C. Napoli, A. Laudani, and G. L. Sciuto, "Optimal thicknesses determination in a multilayer structure to improve the spp efficiency for photovoltaic devices by an hybrid femcascade neural network based approach," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*. IEEE, 2014, pp. 355–362.

[30] J. Guitart, J. Torres, and E. Ayguadé, "A survey on performance management for internet applications," *Concurrency and Comp.: Practice and Experience*, vol. 22, no. 1, pp. 68–106, 2009.

[31] T. Abdelzaher and N. Bhatti, "Web server QoS management by adaptive content delivery," in *Proceedings of IWQoS*. IEEE, 1999, pp. 216–225.

[32] D. Zydek, H. Selvaraj, G. Borowik, and T. Łuba, "Energy characteristic of a processor allocator and a Network-on-Chip," *International Journal of Applied Mathematics and Computer Science*, vol. 21, no. 2, pp. 385–399, 2011.

[33] T. Łuba, G. Borowik, and A. Krasniewski, "Synthesis of finite state machines for implementation with programmable structures," *International Journal of Electronics and Telecommunications*, vol. 55, no. 2, pp. 183–200, 2009.

[34] S. Tambe, A. Dabholkar, and A. S. Gokhale, "CQML: Aspect-oriented modeling for modularizing and weaving QoS concerns in component-based systems," in *Proceedings of ECBS*. IEEE, April 2009, pp. 11–20.

[35] M. Woźniak, W. M. Kempa, M. Gabryel, and R. K. Nowicki, "A finite-buffer queue with single vacation policy – analytical study with evolutionary positioning," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. 887–900, 2014.

[36] M. Woźniak, W. M. Kempa, M. Gabryel, R. K. Nowicki, and Z. Shao, "On applying evolutionary computation methods to optimization of vacation cycle costs in finite-buffer queue," *Lecture Notes in Artificial Intelligence – ICAISC'2014*, vol. 8467, PART I, pp. 480–491, 2014.

[37] M. Woźniak, "On applying cuckoo search algorithm to positioning GI/M/1/N finite-buffer queue with a single vacation policy," in *Proceedings of the 12th Mexican International Conference on Artificial Intelligence - MICAI'2013*. 24-30 November, Mexico City, Mexico: IEEE, 2013, pp. 59–64.