An Approach for the Composition of Generic Cloud-Based Services Using XRI-Based APIs for Enabling New E-Business

Antonio Celesti[1], Francesco Tusa[2], Massimo Villari[3], Antonio Puliafito[4]

DICIEAMA, Università degli Studi di Messina
Contrada Di Dio, S. Agata 98166, Messina, Italia
[1]acelesti@unime.it; [2]ftusa@unime.it; [3]mvillari@unime.it; [4]apuliafito@unime.it

***Abstract*-Nowadays, cloud computing offers more and more business opportunities, and thanks to the concept of virtualization, different types of cost-effective Cloud-based services have been rising. Virtualization of computing, storage, and networking resources, and their interconnection is at the heart of cloud computing, hence enabling new E-Business scenarios. In such a context, APIs for enabling Cloud-based services are strongly required, nevertheless, methods, mechanisms and tools for exploiting virtualized resources and their utilization for developing anything as a service (*aaS) are still ad-hoc and/or proprietary in nature. In this paper, we discuss how to use an adaptive standard protocol, i.e., XRI, for enabling cloud service providers to arrange their own Cloud-based services, building them on top of the IaaS provided by other service providers.**

Keywords- Cloud Computing; Cloud Management; Federation; Service Composition; E-Business

I. INTRODUCTION

Today, cloud computing represents a tempting business opportunity for ICT operators of increasing their revenues [1,2]. The cloud ecosystem begins to be clearer and the role played by cloud service providers appears more defined than the past. Moreover, the number of new public, private, and hybrid clouds rising all over the world is continually growing [3]. The success of cloud computing is due to the fact that it offers new business opportunities for both service providers and their clients (e.g., organizations, universities, ICT societies, other clouds, mobile and desktop customers, etc), by means of architectures for delivering Infrastructure, Platform and Software as a Service (i.e., IaaS, PaaS, and SaaS) [4].

A possible classification of cloud architectures can be made according to a three-tier stack [5]. As depicted in Figure 1, such a stack includes, the following three layers: Virtual Machine Manager (VMM), Virtual Infrastructure Manager (VIM), and Cloud Manager.

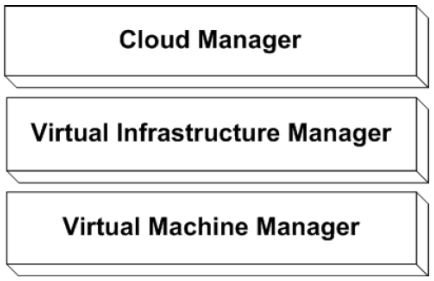

Fig. 1 Three-tier stack describing a cloud architecture

The VMM provides an abstraction for the above layer. It can be a hypervisor (e.g., Xen, KVM, VMware, Virtual Box, Virtual PC, etc) running on top of the Operating System (OS) of each physical server composing the cloud's data centre, and enables the capability of deploying Virtual Machines (VMs).

The VIM acts as a dynamic orchestrator of physical server running hypervisor. It is responsible to arrange IaaS defining which VMs have to be instantiated and in which physical server. The main purpose of this layer is to setup VMs (e.g., preparing disk images, setting up networking, and so on) regardless of the underlying VMM. In the end, the CM implements the business logic of a cloud service provider also addressing security and Quality of Service (QoS). It is responsible to arrange "specific" IaaS, PaaS, or SaaS, related to a target business, on top of "generic" IaaS provided by the underlying layer.

Commonly, these three layers are implemented within an administrative domain and are managed by a single cloud service provider, nevertheless, as predicted by T. Bittman of Gartner [6], the evolution of the cloud computing market is switching from a "Monolithic" (Stage 1), where cloud providers are based on isolated proprietary architectures to a Vertical Supply Chain (VSC) (Stage 2), where cloud providers are able to leverage both Cloud-based services and virtualized resources from other providers in a distributed environment. This scenario brings many advantages for both ICT societies and the whole ICT market, but it raises issues due to the interoperability between the different stack tiers of the cooperating clouds. At the same

time, standard methods, mechanisms, and tools are strongly required for enabling the exploitation of virtualized resources and their utilization for the development of *aaS.

Currently, mega-providers such as Amazon [7] offer IaaS adopting various pay-per-use forms including pay per hour, pay per data transferred, pay per GB-month of provisioned storage, pay per 10.000 GET requests, and so on. Such a form of business allows ICT operators to build their own Cloud-based services on the IaaS, i.e., VMs, supplied by these mega-providers. However, at the time of writing of this paper, there is not a standard methodology which allows binding the Cloud-based services with the IaaS on which they are deployed.

In this paper, we propose a methodology based on a standard protocol, i.e., eXtensible Resource Identifier (XRI) [8], that enables a cloud service provider to develop its own Cloud-based services i.e., PaaS and SaaS, using the virtualized resources, i.e., IaaS, supplied by other providers in a VSC scenario including several cooperating public, private, and hybrid clouds. More specifically, by means of a case of study, we demonstrate as a cloud service provider can manage their composed Cloud-based services as a whole, retrieving information about them from the IaaS on which they are deployed.

The paper is organized as follows: Section 2 describes the state-of-the-art of the existing open source cloud middleware, classifying them according to the three-tier stack. In Section 3, adapting the tree-tier stack for distributed cooperating clouds, we will discuss a scenario of VSC' in which both integration and composition of Cloud-based services take place. In Section 4, we introduce XRI, the protocol that can be adopted for the development of composed Cloud-based services. A concrete use case using XRI is presented in Section 5, highlighting also experimental results. Conclusions are summarized in Section 6.

II. RELATED WORK

As the major commercial cloud providers (e.g., Amazon, Microsoft Azure, Google, Saleforce, Yahoo, etc) adopt proprietary solutions, hiding their inner architectural details, from now on, in this dissemination, in order to describe a VSC scenario from the architectural point of view, we are going to consider open source cloud solutions. This Section describes the current state-of-the-art in cloud computing analysing the main existing open source implementations, evaluating their main features and classifying them according to the three-tier stack. Open source cloud computing solutions include CLEVER [9], OpenQRM [10], OpenNebula [11], OpenStack [12], Nimbus [13], Eucalyptus [14], Claudia [14].

Nimbus is an open source toolkit that allows turning a set of computing resources into an Iaas cloud. Nimbus comes with a component called workspace\--control, installed on each node, used to start, stop VMs and pause, implements VM image reconstruction and management, securely connects the VMs to the network, and delivers contextualization. Nimbus's workspace-control tools work with Xen and KVM but only the Xen version is distributed. Nimbus provides interfaces to VM management functions based on the WSRF set of protocols. There is also an alternative implementation exploiting Amazon EC2 WSDL.

Eucalyptus is an open-source cloud-computing framework that uses the computational and storage infrastructures commonly available at academic research groups to provide a platform that is modular and open to experimental instrumentation and study. Eucalyptus addresses several crucial cloud computing questions, including VM instance scheduling, cloud computing administrative interfaces, construction of virtual networks, definition and execution of service level agreements (cloud-to-user and cloud-to-cloud), and cloud computing user interfaces.

OpenQRM is an open-source platform for enabling flexible management of computing infrastructures. Thanks to its pluggable architecture, OpenQRM is able to implement a cloud with several features that allows the automatic deployment of services. It supports different virtualization technologies managing Xen, KVM and Linux-VServer VEs. It also supports P2V (physical to virtual), V2P (virtual to physical) and V2V (virtual to virtual) migration. This means VEs (appliances in the OpenQRM terminology) can not only easily move from physical to virtual (and back), but that they can also be migrated from different virtualization technologies, even transforming the server image.

OpenNebula is an open and flexible tool that fits into existing data center environments to build a Cloud computing environment. OpenNebula can be primarily used as a virtualization tool to manage virtual infrastructures in the data-center or cluster, which is usually referred as Private Cloud. The more recent versions of OpenNebula are trying to supports Hybrid Cloud to combine local infrastructure with public cloud-based infrastructure, enabling highly scalable hosting environments. OpenNebula also supports Public Clouds by providing Cloud interfaces to expose its functionalities for virtual machine, storage and network management.

OpenStack is IaaS cloud computing project that is free open source software released under the terms of the Apache License. The project is managed by the OpenStack Foundation, a non-profit corporate entity established in September 2012. The technology consists of a series of interrelated projects that controls large pools of processing, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

CLEVER (The CLoud-Enabled Virtual EnviRonment) specifically aims at the design of a VIM layer for the administration of private cloud infrastructures. CLEVER is able to manage cluster of nodes each containing a host level management module

(Host Manager). A single node may also include a cluster level management module (Cluster Manager). All these entities interact exchanging information by means of the Communication System based on the XMPP [16]. The set of data necessary to enable the middleware functioning is stored within a specific Database deployed in a distributed fashion. CLEVER offers security and fault-tolerance and thanks to its modular and pluggable architecture represent a multi-platform solution integrating with lots of software solutions.

Commonly, VIM layer middleware are able to use at VMM layer at least one third party hypervisor solution. Neglecting the VMM layers, Table 1 highlights which open source solutions match the VIM and CM layers.

TABLE I CLASSIFICATION OF OPEN SOURCE CLOUD MIDDLE WARES ACCORDING TO THE THREE-TIER STACK

Middleware	VIM	CM
Nimbus	x	x
Eucalyptus	x	x
OpenQRM	x	
OpenNebula	x	
OpenStack	x	
CLEVER	x	
CLAUDIA		x

In a scenario of hybrid clouds, projects such as Nimbus and Eucalyptus, which can be considered both at CM and VIM layers, manage VMs themselves directly, without providing the full set of features typical of the VIM layer. On the other hand, projects such as OpenQRM, OpenNebula, and CLEVER acting at VIM layer provide better virtual infrastructure management tools, but do not have CM features. In fact, they are able to manage VMs, either individually or in groups that need parallel scheduling on local resources or external public clouds. In addition, they automate VM setup (preparing disk images, setting up networking, and so on) regardless of the underlying VMM layer. In the end, projects such as CLAUDIA are able to manage as a whole cloud services, nevertheless they merely act at CM layer and need to interact with other solution acting at VIM layer.

III. HOW TO ENABLE A CLOUD TO ARRANGE CLOUD-BASED SERVICES USING MULTIPLE THIRD PARTY IAAS (S)

The delivery of complex services, traditionally enhanced by service providers in client/server architecture, has become a necessity of our society. One outstanding problem of these architectures is that the demand of services is very difficult to predict: for this reason, resources are generally over provisioned in order to support peak demands. However, this implies high costs for both service providers and consumers. Besides, the current adopted solutions still need to find new heuristics for the delivery of ubiquitous, continuous and reliable services. With the advent of cloud computing, ICT operators have been aiming to overcome these problems. However, Cloud Service Providers to cost-efficiently deliver capacity, need mechanisms enabling reliable management of cloud-based services, and that efficiently manage all phases of the service life cycle. To simplify service deployment, an easy-to-use, comprehensive mechanism that allows the description, composition, installation, and configuration of complex Cloud-based services is required [17,18,19].

A. Vertical Supply Chan From an Architectural Point of View

In general, considering the three-tier stack, we could say that the Cloud Manager layer implements the business logic of a Cloud Service Provider. In order to achieve a VSC, we generalized the three-tier stack for meeting the requirement of a business cloud ecosystem in which the Cloud Service Providers "cooperate" on the base of business agreements in order to compose Cloud-based services and deploying them on the IaaS supplied by other providers. An example of Vertical Supply Chain is depicted in Figure 2.

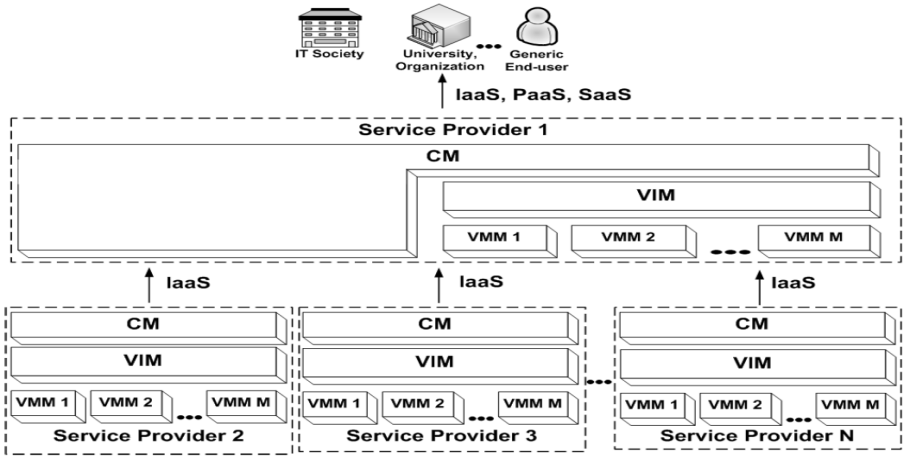

Fig. 2 Three-tier stack describing a cloud architecture

Service Provider 1 is responsible to arrange a "specific" Cloud-based service, on top of "generic" IaaS(s) provided by the underlying layer. The computing infrastructure acting at the VIM layer may belong to Service Provider 1 itself or may be offered by third-party Cloud Providers 2, 3, ..., N. In the scenario pointed out in Figure 2, requests for service allocation received from Service Provider 1 are processed and eventually IaaS instantiation requests are forwarded to Service Providers 2, 3, ... , N, where are caught by the CM layer (which acts as interface between the "world" and the infrastructure) and sent to the underlying VIM layer. This latter will offer the ability of deploying VMs on which the IaaS, PaaS, and SaaS will be executed according to a dynamic and scalable fashion: each VIM of Cloud Providers 2, 3, ..., N, in order to satisfy the service allocation coming from the overlying CM, will exploit the capabilities of specific hypervisors acting at VMM layer, thus creating an abstraction for the physical datacenter(s).

Considering the open source middleware analysis of Section 2, it is obviously that this scenario well fits middleware acting at CM layer such as Claudia and middleware acting at VIM layer such as OpenNebula, OpenQRM, and CLEVER. However, this scenario can be applied also to cloud middleware acting both at CM and VIM such as Nimbus and Eucalyptus, even though, how has been previously discussed, these latter do not offer so many features as well as middleware specialized for the VIM layer. Moreover, it is obvious that this scenario can be applied also to commercial IaaS provider such as Amazon.

The scenario we are considering might be used to address situations in which a Cloud Service Provider acting both at CM and VIM layers has expired its computational capabilities, and it is not able to satisfy further service requests anymore. Such a scenario is depicted in Figure 3.

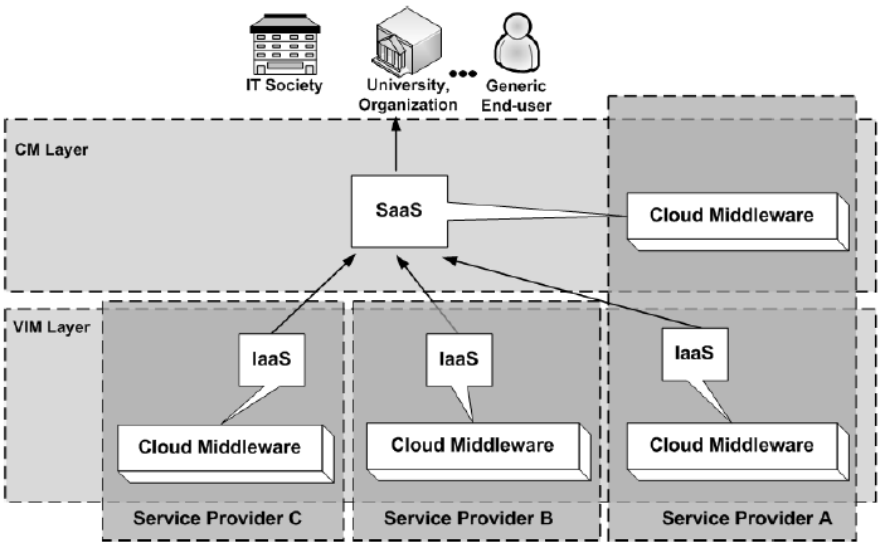

Fig. 3 Service composition in a VSC scenario

Cloud Provider A acts both at CM and VIM layers offering IaaS, PaaS, and SaaS. It wants to build a Cloud-based SaaS on top of a IaaS, but it does not have enough resources to arrange the required IaaS. Thus, in order to satisfy the SaaS allocation requests, it contacts respectively Cloud Service Providers B and C. Therefore, the SaaS will be built on top of the IaaS provided by Provider A datacenter also using the IaaS(s) provided respectively by Service Provider B and C. Nevertheless, this leads to a complex management of the resources because the running services will be deployed on independent computing infrastructures.

B. Vertical Supply Chain Issues

The scenario we are pointing out is more complex than a traditional "monolithic" one, as a smart, improved coordination system is needed for enabling the cooperation of the different involved Cloud Service Providers. In this new cloud perspective, together with the traditional internal monitoring mechanisms of each Service Provider, the service metering information became important both for accounting, billing and service placement decisions among the involved Service Providers: through this approach, each Service Provider is able to achieve an efficient use of the hardware resources, consequently minimizing costs.

More specifically, as depicted in Figure 3, when the hardware resources of one Service Provider are insufficient, or it is necessary to re-arrange the service distribution for meeting the Service Level Agreement (SLA) scheduled with the Clients, the CM layer of a Cloud can exploit the resources coming from the VIM layers of other Service Providers Providers. Considering Figure 3, the CM and VIM layers of Service Provider A may be implemented by the same software solution (e.g., using Nimbus or Eucalyptus) or integrating a software solution acting at VIM layer (e.g., OpenNebula, OpenQRM. OpenStack, or CLEVER) with other software solutions acting at CM layer (e.g., CLAUDIA).

In order to achieve and implement this scenario, some issues have to be addressed and some new features have to be

included within the middleware. The software solution of CM layer, in fact, must be able to interact with the underlying VIM middleware regardless of their implementation, in order to monitor the state of the computing infrastructure and consequently send specific commands to them. In particular:

- it should be able to compose and instantiate Cloud-based services deploying them on the IaaS supplied other service providers by means of standard APIs;
- it needs a specific mechanism for tracing the state of the requested Cloud-based services and maintain information about where they are deployed (i.e., the specific VIM middleware on which a service is deployed);
- it should be able to access the VIM middleware services using a "common interface" that allows sending generic commands to the VIM (e.g., start, stop and destroy VMs);
- in order to know the resource allocation state on each VIM middleware, a unified method for describing such an information has to be employed;

A solution able to fulfill these requirements in a cloud scenario has not been defined yet and finding it is not trivial at all.

IV. THE XRI TECHNOLOGY

In this Section, after a brief description of the XRI protocol, we motivate how it can be used for enabling the arrangement of Cloud-based services in a VSC scenario.

A. XRI and XRDS

The XRI protocol provides a standard syntax for identifying entities, regardless any particular concrete representation. The XRI system is similar to DNS, including a set of hierarchical XRI authorities but more powerful. The protocol is built on URI (Uniform Resource Identifiers) and IRI (Internationalized Resource Identifiers) extending their syntactic elements and providing parsing mechanisms. Particular types of URI are URN and URL. Since an URL is also an URI, the protocol provides a parsing mechanism from XRI to URL. Therefore XRI is also compatible with any URN domain.

XRI supports persistent and reassignable identifiers by means of i-numbers (Canonical ID) and i-names (Local ID). It also provides four types of synonyms (LocalID, EquivID, CanonicalID, and CanonicalEquivID) to provide robust support for mapping XRIs, IRIs, or URIs to other XRIs, IRIs, or URIs that identifies the same target entity. This is particularly useful for discovering and mapping to persistent identifiers as often required by trust infrastructures. XRI enable organization to logically organize entities building XRI tree. According to the XRI terminology, each entity in the tree is named authority. The protocol provides two additional options for identifying an authority: Global Context Symbols (GCS) and cross-references. Common GCS are "=" for people, "@" for organization, and "+" for generic concepts. For example the xri://@XYZ*marketing indicates the marketing branch of an organization named XYZ, where the "*" marks a delegation. Instead, cross-reference allows XRI references to contain any other XRI references or IRIs as sub-segments. An example is xri://@XYZ*(xri://ABC*development) indicating that the development branch of the XYZ organization is managed by the ABC organization.

An authority is resolved by means of an XRDS document representing a simple, extensible XML resource description format standard describing the features of any URI, IRI, or XRI-identified entity in a manner that can be consumed by any XML-aware system. Each XRDS describes which types of information are associated to an authority and the way in which they can be obtained. Using HTTP, XRI resolution involves two phases: authority resolution which is the phase required to resolve an XRI into a XRDS document from an XRI Authority Resolution Server (ARS), and Service End-Point Selection which is the phase of selection of the SEP server (e.g., web service, service provider, web application) returning the data describing the entity in a given context. The same SEP server can also return different data of the same authority.

B. Why Does XRI suit to Cloud Computing?

In our opinion, XRI can be adopted to develop a seamless mechanism for the composition of Cloud-based services in a VSC scenario. As XRI is compatible with IRI naming systems, there is not the need to use a unique global naming system, even though this would be possible. This feature allows clouds to manage their own XRI naming systems, mapping them on the global DNS maintaining the compatibility with the existing naming systems. Moreover, with XRI a cloud can keep one tree representing IaaS, PaaS, and SaaS. In addition, such a technology can be used for both identify and resolve VMs and whole *aaS by means of the resolution of XRI authorities. Moreover, the XRDS document can be used to describe an XRI authority associated to a target service of VM, indicating how to resolve it by means of the corresponding SEP.

For example the cloud service provider may need to retrieve three types of information about an authority representing a VM, resolving it in three different ways. In the first way, the VM has to be resolved by means of general data (e.g., CPU, memory, kernel, operating system); in the second way, the VM has to be resolved by means of real time performance data (e.g., amount of used CPU and memory used); instead in the third way, the VM has to be resolved by means real time data regarding an internal running application (e.g., the percentage of processed data). Such a situation can be addressed by mean of three different XRDs inside the XRDS document corresponding to the XRI authority associated to the VM, each one pointing to a target SEP server. Figure 4 depicts an example of authority resolution performed by Service Provider A in Service Provider B.

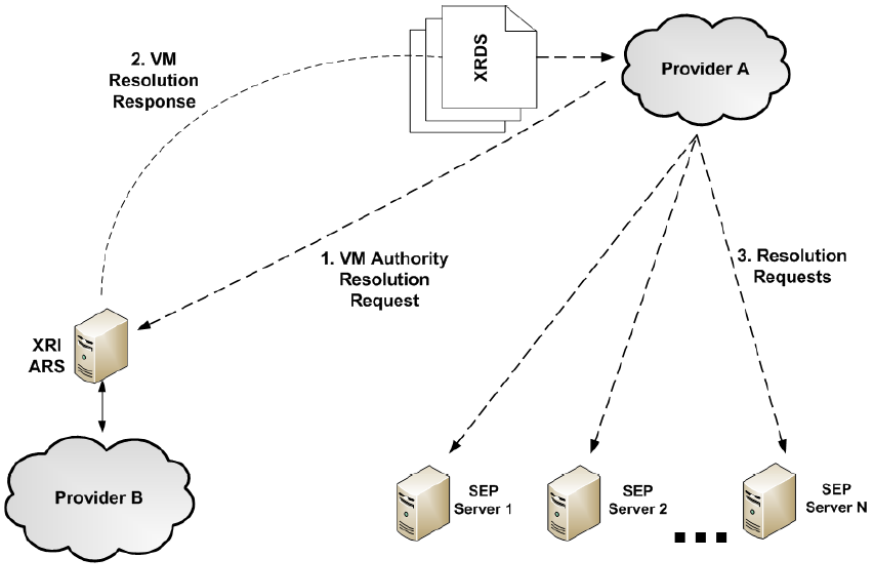

Fig. 4 Resolution of data associated to a VM using XRI

In Step 1 the cloud Service Provider A performs a resolution request of an authority representing a VM to the XRI ARS of Service Provider B. In Step 2 the XRI ARS responses with a XRDS document describing the target VM. In Step 3 the Service Provider A analyzes the XRDS and find out how to retrieve the required information. Thus, the Service Provider A queries the target SEP server and obtains the required information.

V. CLOUD-BASED SERVICE ARRANGEMENT USING XRI-BASED INTERFACES

In this Section, after a description on how the XRI technology may be used for addressing the issues debated in the previous, using the XRI ARS provided by the OpenXRI project [20], we are going to simulate the XRI operations needed in a Vertical Supply Chain scenario for the composition of Cloud-based services, evaluating their performances.

A. Use Case Overview

In order to clarify the ideas on how the XRI protocol can help the composition, development, and management of Cloud-based services, we are going to consider a possible concrete business scenario involving a Service Provider and third party Service Providers able to leas resources. Let us suppose that a Cloud Service Provider named “Video Cloud SP” supplies a SaaS (e.g., video transcoding, video streaming, etc), building it on top of an IaaS consisting of several VMs running within its virtualization infrastructure, managed by a VIM middleware. Using the offered SaaS application, a Client will be able to exploit a high-level performance transcoding service because the Video Cloud SP (behind the scenes) parallelizes the video transcoding process, distributing the video source data to be processed among different VMs.

Let us suppose that the Video Cloud SP runs out of its virtualization resources: in order to carry on providing video transcoding SaaS, it may exploit a Vertical Supply Chain using the IaaS(s) offered by external Service Providers. As previously stated in Section \ref{subsec:issues}, several issues have to be overcome. As depicted in Figure \ref{fig:video-cloud-example}, the video transcoding SaaS is built on top of three different IaaS: the first one is provided by the Virtualization Infrastructure of Video Cloud SP itself, whereas the second one and the third one are respectively supplied by the Virtualization Infrastructures of Providers B and C. This implies that, besides the information about its own IaaS, the Video Cloud SP must be also aware about the status of the other IaaS(s). E.g., for each IaaS, it has to know how many VMs have been arranged and the percentage of completed tasks within each of them.

B. XRI Data Retrieval for the Management of Cloud-Based Services

The Video Cloud SP needs a method for describing the logical organization of its own running services and the different instances: this task can be accomplished using the XRI tree structure.

As depicted in Figure 5, in order to logically manage their Cloud-Based services and retrieve associated information, Video Cloud SP, Provider B and Provider C use their own XRI ARSs and several SEP. More specifically, Video Cloud SP is represented by the XRI authority *xri://@VideoCloudSP*, whereas at the underlying level are mounted several authorities including **v_trans* and **v_str* respectively representing video transcoding and video streaming services. Moreover, considering the XRI authority **v_trans* the underlying mounted authorities identify the different instances for that service. In turn, each instance may be built on top of one or more IaaS(s). For example, the instance represented by the XRI authority

inst_1 is arranged on three different IaaS(s): one hosted in the Virtualization Infrastructure of the Video Cloud SP itself, whereas the others are placed in the external Service Providers B and C.

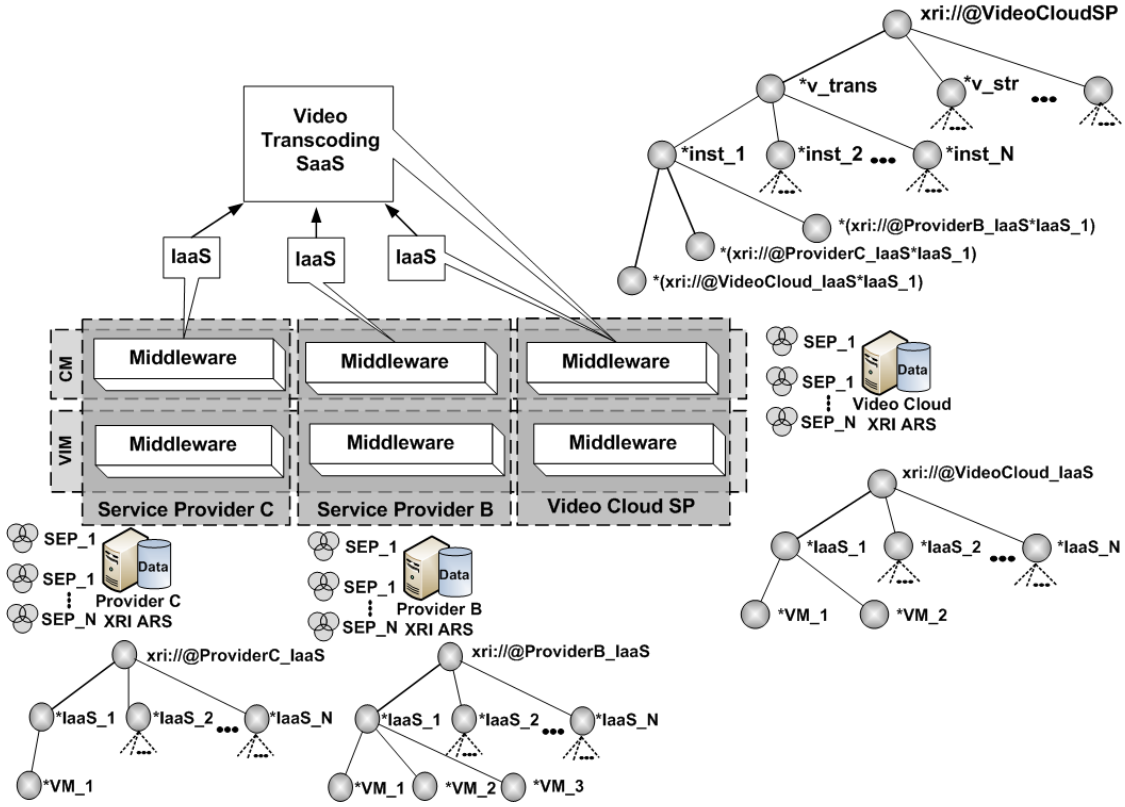

Fig. 4 Video Cloud Service Provider using External Virtualization Infrastructures

As for the SaaS(s) of Video Cloud SP, Provider B, Provider C, and Video Cloud Itself, logically organize their IaaS(s) mapping them on their own XRI trees. The Video Cloud SP, using its own XRI ARS, builds another XRI tree, with *xri://VideoCloud_IaaS* authority root, representing its IaaS(s). At the underlying level are mounted the authorities representing the different IaaS instances, e.g., the authority *IaaS_1* describes the used resources, and the underlying mounted authorities **VM_1* and **VM_2* represents the VMs on which the IaaS instance is deployed. More specifically, the paths to resolve the two VMs respectively are:

- *xri://VideoCloud_IaaS*VM_1*
- *xri://VideoCloud_IaaS*VM_2.*

Similarly, Providers B and C logically organize their own IaaS instances using their own XRI ARSs: e.g., in example, the IaaS instance of Provider B, i.e., *xri://ProviderB_IaaS*IaaS_1* is deployed on three different VMs represented by the XRI authorities:

- *xri://ProviderB_IaaS*IaaS_1*VM_1*
- *xri://ProviderB_IaaS*IaaS_1*VM_2*
- *xri://ProviderB_IaaS*IaaS_1*VM_3.*

In the same way, considering Provider C, the authority identified by *xri://ProviderC_IaaS*IaaS*VM_1* represents the VM on which the *xri://ProviderC_IaaS*IaaS_1* is deployed.

The advantage of using XRI is that it allows creating an abstraction from the implementation of the CM and VIM layers. Considering the aforementioned video transcoding SaaS(s) the Video Cloud SP, at CM layer, needs a method to manage the instance identified by the *xri://@VideoCloudSP*inst_1* authority. More specifically, the Video Cloud SP has to know the IaaS(s) on which the instance of the video transcoding SaaS is deployed (both its own IaaS and the IaaS(s) supplied by Providers B and C). Thanks to XRI, this is possible using cross-reference mechanisms, mounting under the *xri://@VideoCloudSP*inst_1* the cross-reference authorities:

- **(xri://VideoCloud_IaaS)*
- **(xri://ProviderB_IaaS*IaaS_1)*
- **(xri://ProviderC_IaaS*IaaS_1).*

In this way, resolving these cross-reference authorities by means of appropriate SEPs, the Video Cloud SP is able to know the VMs where each IaaS is deployed, retrieving for each VM information such as the IP address, the percentage of used CPU, the amount of used memory, the amount of used internal storage, and the status of the running process associated to the SaaS (i.e., in our example the percentage of completed transcoded fragment of video). Thanks to such information, the Video Cloud SP will be able to build and update in real time the web graphical interface for the end-user of the video trascoding SaaS.

Figure 6 depicts the resolution process performed by the Video Cloud SP which resolves the XRI authority associated to the IaaS supplied by Provider C.

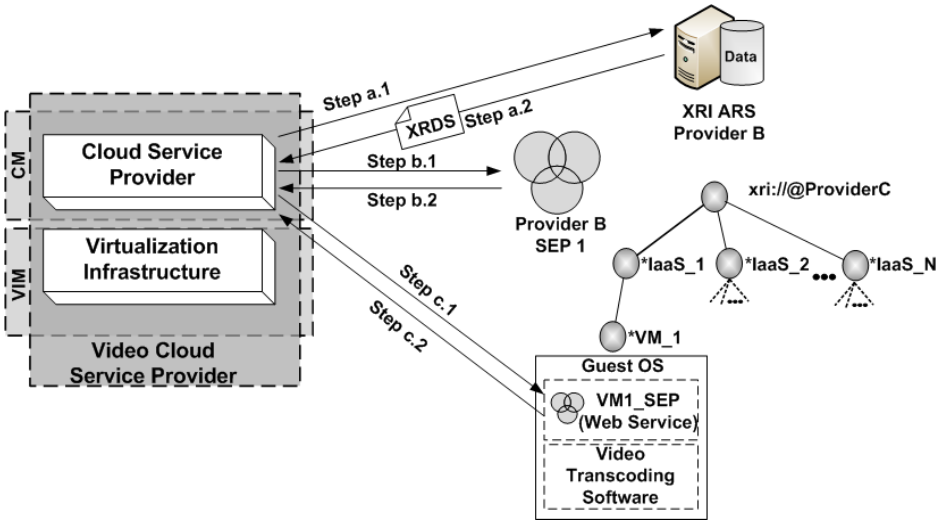

Fig. 5 Video Cloud Service Provider using External Virtualization Infrastructures

In Step a.1 the Video Cloud SP resolves the authority *xri://ProviderC_IaaS*IaaS_1*. In Step a.2 the XRI ARS of Provider C responses with the corresponding XRDS document describing the target IaaS. The SP reads the XRDS and discovers the different SEPs returning particular data associated to the IaaS. In Step b.1 the SP selects the SEP1 of physical Provider C returning general information about the IaaS. SP wants to find out the information associated at the VMs where the IaaS is deployed and therefore it needs to discover the XRI authorities associated to them. This task can be achieved in two different ways according to the implementation: 1) recursively passing through the XRI tree starting from the **IaaS_1* authority; 2) retrieving this information from the XRDS document obtained at Step a.2. Considering this latter possibility, in Step c.1, the SP discovers that the IaaS is deployed on one VM whose XRI authority can be resolved by means of the path *xri://ProviderC_IaaS*IaaS_1*VM_1*. In Step c.2 the SP obtain the corresponding XRDS. As the SP wants to obtain from the VM the information regarding the real time percentage of completed transcoded fragment of video, it discovers that in this particular case the associated SEP is placed in the same VM in form of web service. In this way the SP is able to manage the whole video transcoding SaaS.

The use of a SEP placed inside the VM allows a cloud service provider to flexibly manage composed services without the need to apply changes to the physical hardware composing the XRI infrastructure. In fact, for Cloud Providers the physical SEP describing general IaaS(s) is always the same, instead the SEP describing specific VMs can be customized and embedded within them when the disk-images are prearranged for a target IaaS.

C. Cross-Mounting Performance Evaluation

In this Section we are going to analyse the experimental results collected from a set of simulation that we performed using the ARS made available from the open source project OpenXRI. Our idea was to reproduce an environment able to fit the composition of Cloud-based services through XRI-based interfaces in the VSC scenario, considering different arrangements of the XRI authorities, even reproducing “extreme” functioning conditions.

The series of tests executed (50 runs for each simulation, performed using the JMeter tool [21]) guarantee a wide coverage of possible results. The confidence interval (at 95%) depicted in all graphs indicates the goodness of our analysis. Our testbed has been deployed inside a system running a Redhat Enterprise Linux AS Release 3.0.8 operating system having the following hardware features: Blade LS21 AMD Opteron Biprocessor Dual Core 2218 2.6GHz 8GB RAM.

To perform this analysis and estimate the workload of OpenXRI ARS 1.2.1 we submitted cross-mounting requests using JMeter and evaluated the Response Time of the system expressed in *msecs*. We have measured the time interval between the request phase to the system at *Ts* and the response time at *Tr* taking place in the receiving phase.

In our graphs we reported the total time spent to accomplish each task: $Tt=Ts+Tr$. The exchange of requests and responses is measured in a local network (LAN, without any Internet connection), since the measurements are not affected from the

network communication parameters (e.g., throughput, delays, jitter, etc).

All the collected results have been statistically analysed and presented schematically in Figures 6 and 7. These latter consider the response time of cross-mounting tasks in both wide and deep XRI tree structures. Figure 6 depicts the response time of authority cross-mounting tasks considering a wide tree structure. On the x-axis we have represented the number of XRI authorities, whereas on the y-axis we have represented the response time expressed in milliseconds. For this experiment, we have response times that range from about 1 second to less of 2 seconds considering 10, 100, and 1000 authorities, a result very good considering cloud computing environments.

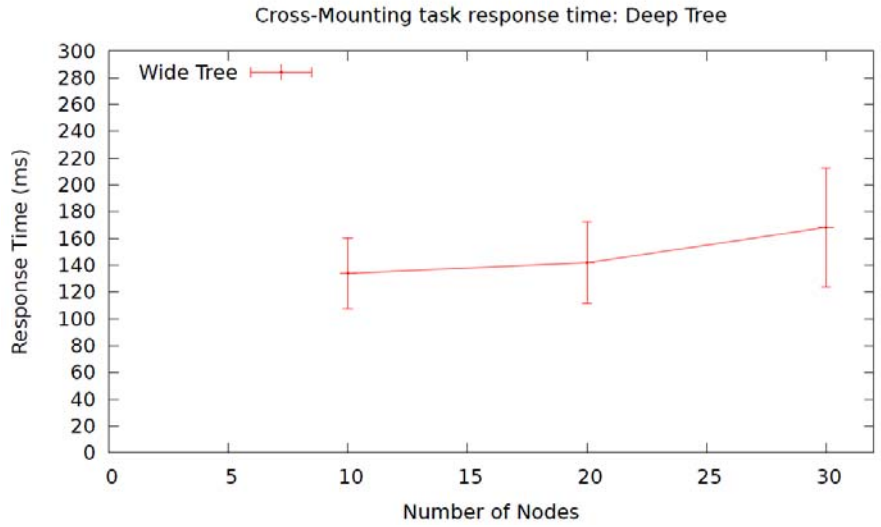

Fig. 6 Video Cross-Mounting task response time: "Wide Tree"

In the end, Figure 6 depicts the response time trend of authority cross-mounting tasks considering the deep tree structure. On the x-axis we have represented the number of levels of the XRI tree structure, whereas on the y-axis we have represented the response time expressed in milliseconds. In such an experiment, considering 10, 20, and 30 tree levels, we can observe how the response time is indeed negligible: less of 0.2 seconds.

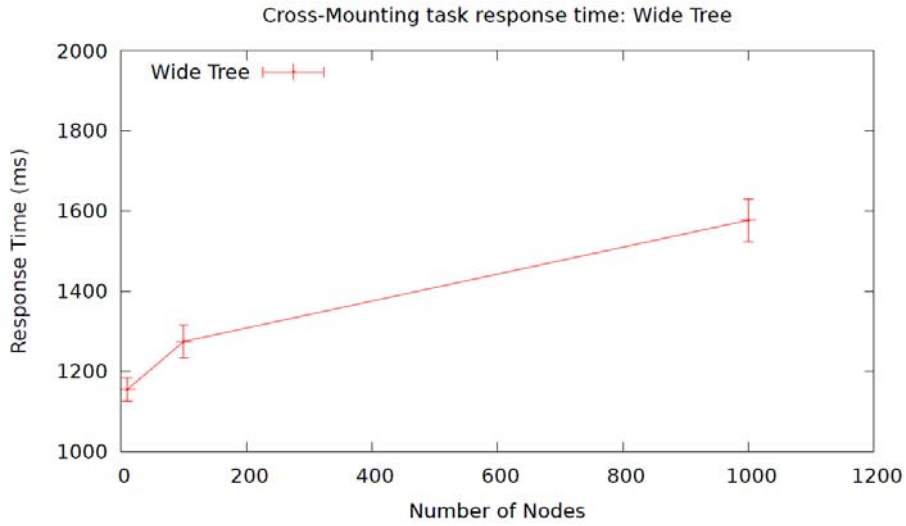

Fig. 7 Cross-Mounting task response time: "Deep Tree"

For the aforementioned reasons we can state that, in the OpenXRI implementation, the cross-mounting tasks are not dependent from the structure of the tree (wide or deep).

VI. CONCLUSION

In this paper we tackled the composition of Cloud-based services in VSC scenarios considering hybrid cloud built according to a tree-tier stack. More specifically, we focus on a scenario including a cloud service provider which composes Cloud-based services deploying them the IaaS provided by other providers. In order to manage such services, a cloud needs to retrieve data about the IaaS(s) on which the services are deployed. In order to accomplish this task a solution using the XRI

protocol and its cross-reference mechanism has been discussed. Moreover, considering the XRI ARS provided by the OpenXRI project several experiments have been made. Information retrieval about composed Cloud-based services is only one of the possible issues in the VSC topic. We hope we succeed to stimulate your interest in further contributing about such a topic.

ACKNOWLEDGMENT

The research leading to the results presented in this paper has received funding from the European Union's Seventh Framework Programme (FP7 2007-2013) Project RESERVOIR under grant agreement number 215605 and from the Union's Seventh Framework Programme (FP7 2007-2013) Project VISION-Cloud under grant agreement number 217019.

REFERENCES

[1] H. Cai, K. Zhang, M. Wang, J. Li, L. Sun, and X. Mao, "Customer centric cloud service model and a case study on commerce as a service," in Proceedings of the 2009 IEEE International Conference on Cloud Computing, CLOUD '09, (Washington, DC, USA), pp. 57–64, IEEE Computer Society, 2009.

[2] Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," Cloud Computing, IEEE International Conference on Cloud Computing, pp. 123–130, 2010.

[3] Sun Microsystems, Take your business to a Higher Level - Sun cloud computing technology scales your infrastructure to take advantage of new business opportunities, guide, April 2009.

[4] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in Grid Computing Environments

[5] Workshop, 2008. GCE '08, pp. 1–10, 2008.

[6] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," Internet Computing, IEEE, vol. 13, pp. 14–22, September 2009.

[7] T. Bittman, "The evolution of the cloud computing market," Gartner Blog Network, http://blogs.gartner.com/thomas bittman/2008/11/03/theevolution-of-the-cloud-computing-market/, November 2008.

[8] Amazon Elastic Compute Cloud (Amazon EC2): http://aws.amazon.com/ec2/.

[9] Extensible Resource Identifier (XRI) Syntax V2.0, Committee Specification, OASIS, 2005.

[10] F. Tusa, M. Paone, M. Villari, and A. Puliafito., "CLEVER: A CLoud-Enabled Virtual EnviRonment," in 15th IEEE Symposium on Computers and CommunicationsS Computing and Communications, 2010. ISCC '10. Riccione, June 2010.

[11] OpenQRM, "the next generation, open-source Data-center management platform", http://www.openqrm.com/.

[12] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," in High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on, pp. 59–68, June 2009.

[13] OpenStack, "Open source software for building private and public clouds", http://www.openstack.org.

[14] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the Use of Cloud Computing for Scientific Workflows" in SWBES 2008, Indianapolis, December 2008.

[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud Computing System," in Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, pp. 124–131, May 2009.

[16] MORFEO Claudia, http://claudia.morfeoproject.org/wiki/index.php.

[17] Extensible Messaging and Presence Protocol (XMPP), http://xmpp.org.

[18] T. V. Pham, H. Jamjoom, K. Jordan, and Z.-Y. Shae, "A service composition framework for market-oriented high performance computing cloud," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, (New York, NY, USA), pp. 284–287, ACM, 2010.

[19] J. O. Gutierrez-Garcia and K.-M. Sim, "Self-organizing agents for service composition in cloud computing", in Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, CLOUDCOM '10, (Washington, DC, USA), pp. 59–66, IEEE Computer Society, 2010.

[20] K. Kofler, I. u. Haq, and E. Schikuta, "User-centric, heuristic optimization of service composition in clouds," in Proceedings of the 16th international Euro-Par conference on Parallel processing: Part I, EuroPar'10, (Berlin, Heidelberg), pp. 405–417, Springer-Verlag, 2010.

[21] OpenXRI Project, XRI applications and libraries, http://www.openxri.org.

[22] Jmeter, the graphical server performance testing too, http://jakarta.apache.org/jmeter.

Antonio Celesti was born in Messina on April 30th 1984. He received the Master Degree in Computer Science at University of Messina (Italy), with the final score 110/110 cum summa laudae. In 2012, he received the PhD in "Advanced Technology for Information Engineering" at the University of Messina (Italy) defending the thesis "Security, Resource Provisioning, and Information Retrieval in Federated Cloud Computing". From 2008 he performs teaching and research activities at the University of Messina. From 2008 he is also one of the members of the Multimedia and Distributed Systems Laboratory (MDSLab).

He was assistant professor of Foundations of Computer Science (2009), Computer Science Laboratory (2009), Computer Networks (2010), Network Security (2010), Calculators (2011), and Operating Systems (2011). In 2012,

he was lecturer of System Security and Information Systems. From 2012 is Assistant Researcher at the Faculty of Engineering of the University of Messina (Italy). His scientific activity has been focused on studying distributed systems and cloud computing. His main research interests include cloud federation, services, information retrieval and security.

Dr. Antonio Celesti won the best paper award at the Second International Conference on Advances in Future Internet, Venice, Italy, in 2012 and the best paper award at the Third International Conference on Evolving Internet, held in Luxembourg in 2011.

Francesco Tusa was born in Messina on February 5th 1983. He received the Master Degree in "Computer Engineering" from the University of Messina (Italy) in November 2007, with the final score 110/110 cum summa laudae. He defended the thesis "Security in Grid environments: enabling data cryptography through smart cards". In 2008, he received a Master Postdegree in "Open Source and Computer Security", and started his PhD studies in "Advanced Technologies for Information Engineering" at the University of Messina. On April 2011 he discussed his PhD thesis "Security in distributed computing systems: from Grid to Cloud".

From April 2007 to October 2008 he worked at the University of Messina within the PON Project "Progetto per l'Implementazione e lo Sviluppo di una e-Infrastruttura in Sicilia basata sul paradigma della grid (PI2S2)" with the aim of design and develop security solutions for Grid environments. He has been actively working as IT Security and Distributed Systems Analyst in cloud computing, virtualization and Storage for the European Union Projects "RESERVOIR" and "VISION-CLOUD". He is involved in the design and implementation of the CLEVER cloud middleware. His scientific activity has been focused on studying distributed systems, grid and cloud computing. His research interests are in the area of security, virtualization, migration, federation of distributed computing systems. He is one of the members of the mdslab.

Massimo Villari was born in Messina on 25th May 1972. In 2003, he got his PhD in Computer Engineering. Since 2006 he is an Aggregate Professor at University of Messina.

He is actively working as IT Security and Distributed Systems Analyst in cloud computing, virtualization and Storage for the European Union Project "VISION-CLOUD" and the previous EU initiative "RESERVOIR". Previously, he was an academic advisor of STMicroelectronics, helps an internship in Cisco Systems, in Sophia Antipolis, and worked on the MPEG4IP and IPv6-NEMO projects. He investigated issues related with user mobility and security, in wireless and ad hoc and sensor networks. He is IEEE member. Currently he is strongly involved on EU Future Internet initiatives, specifically Cloud Computing and Security in Distributed Systems. His main research interests include virtualization, migration, security, federation, and autonomic systems. Cloud Architect @ UniMe for the development of a cloud middleware, named CLEVER.In UniMe, Engineering Faculty, he teaches Java Object Oriented Programming and Database courses.

Antonio Puliafito is a full professor of computer engineering at the University of Messina, Italy. His interests include parallel and distributed systems, networking, wireless, GRID and Cloud computing. During 1994-1995 he spent 12 months as visiting professor at the Department of Electrical Engineering of Duke University, North Carolina - USA, where he was involved in research on advanced analytical modelling techniques. He is the coordinator of the Ph.D. course in Advanced Technologies for Information Engineering currently available at the University of Messina and the responsible for the course of study in computers engineering. He was a referee for the European Community for the projects of the fourth, fifth and sixth Framework Program and he is currently acting as a referee also in the seventh FP.

Dr. Puliafito is co-author (with R. Sahner and Kishor S. Trivedi) of the text entitled "Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package". He is currently the director of the RFIDLab, a joint research lab with Oracle and Intel on RFID and wireless. From 2006 to 2008 he acted as the technical director of the Project 901, aiming at creating a wireless/wired communication infrastructure (winner of the CISCO innovation award). He is currently a member of the general assembly and of the technical committee of the FP7 Vision project.