Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds

Michael Boniface, Bassem Nasser, Juri Papay, Stephen C. Phillips, Arturo Servin, Xiaoyu Yang, Zlatko Zlatev
IT Innovation Centre
University of Southampton
Southampton, UK
e-mail: {mjb,bmn,jp,scp,als,kxy,zdz}@it-innovation.soton.ac.uk

Spyridon V. Gogouvitis, Gregory Katsaros, Kleopatra Konstanteli, George Kousiouris, Andreas Menychtas, Dimosthenis Kyriazis
Telecommunications Laboratory
National Technical University Athens
Athens, Greece
e-mail: {sgogouvitis, gkats, kkonst,ameny,dkyr}@telecom.ntua.gr

***Abstract* - Cloud computing offers the potential to dramatically reduce the cost of software services through the commoditization of information technology assets and on-demand usage patterns. However, the complexity of determining resource provision policies for applications in such complex environments introduces significant inefficiencies and has driven the emergence of a new class of infrastructure called Platform-as-a-Service (PaaS). In this paper, we present a novel PaaS architecture being developed in the EU IST IRMOS project targeting real-time Quality of Service (QoS) guarantees for online interactive multimedia applications. The architecture considers the full service lifecycle including service engineering, service level agreement design, provisioning and monitoring. QoS parameters at both application and infrastructure levels are given specific attention as the basis for provisioning policies in the context of temporal constraints. The generic applicability of the architecture is being verified and validated through implemented scenarios from three important application sectors (film post-production, virtual augmented reality for engineering design, collaborative e-Learning in virtual worlds).**

Keywords -cloud computing; service-oriented infrastructures; platform-as-a-service; quality of service; real-time

I. INTRODUCTION

Cloud computing is one of the hottest buzzwords in information technology today. Through the virtualization of hardware, rapid self-service provisioning, scalability, elasticity, accounting granularity and cost allocation models, Clouds promise the ability to efficiently adapt resource provisioning to the dynamic demands of Internet users. This paper describes a novel Platform-as-a-Service architecture being developed in the European Commission supported IRMOS project [1]. The architecture aims to provide tools and techniques for modelling, simulating, analyzing, planning, provisioning and monitoring real-time service-oriented applications deployed within clouds of virtualized computing, storage and networking where a guaranteed QoS is needed.

This paper reviews the emerging Cloud marketplace in relation to the Service/Platform/Infrastructure (SPI) layered model focusing on PaaS characteristics and support for QoS guarantees. Requirements for real-time multimedia applications are then presented, alongside the application scenarios used to verify and validate the generic applicability of the architecture across different business sectors. All architectural concepts are then described combining service-oriented patterns with real-time modelling techniques.

II. CLOUDS AND REAL-TIME QOS

Cloud computing is a generalized paradigm; therefore it is impossible to consider 'the cloud' as a single set of business models with a single set of Quality of Service issues. To some extent, issues with cloud computing are necessarily related to application characteristics and purpose. However, it is possible to identify cloud types, common stakeholders and their concerns. Today, there are three main classes in the cloud services stack which are generally agreed upon:

- Infrastructure as a service (IaaS): the provision of 'raw' machines (servers, storage, networking and other devices) on which the service consumers install their own software, usually as virtual machine images.
- Platform as a service (PaaS): the provision of a development platform and environment providing services and storage, hosted in the cloud.
- Software as a service (SaaS): the provision of a pre-defined application as a service over the Internet or distributed environment.

A major challenge for SaaS providers wanting to exploit the benefits of cloud computing is to manage QoS commitments to customers throughout the lifecycle of a service. The complexity of this problem has driven the emergence new PaaS offerings that aim to abstract this complexity through targeted tools and services. PaaS aims to be a developer's friend. The idea is simple, even if the execution is complex: multiple applications share a single development platform and common services, including authentication, authorization, and billing. PaaS developers build web applications without installing any tools on their computer and deploy those applications without needing to know or care about the complexity of buying and managing the underlying hardware and software layers. A PaaS is built on an IaaS and uses a multi-tenanted deployment and development tools. A good example of PaaS is Facebook [3],

a venue where multiple applications share resources and user information, subject to tight controls. PaaS stakeholders include:

- the PaaS hoster: must provide adequate resources (typically via an IaaS model) in order to meet demands of its customers' needs, together with appropriate availability contingencies.
- the PaaS provider: will provide an environment suitable for general developers to build web applications without deep domain expertise of back-end server and front-end client development or website administration.
- the PaaS user (developer): must have a browser-based development environment, the ability to deployment seamlessly to a hosted runtime environment, management and monitoring tools and pay as you go billing.

Many PaaS providers exist today such as Google AppEngine, Microsoft Azure, Salesforce.com Force.com, Rackspace Sites, Bungee Connect, EngineYard, Heroku, Intuit, Cloudera, Aptana, VirtualGlobal, LongJump, AppJet, Wavemaker, Aprenda, etc. As far as we can ascertain none of these PaaS providers offer generalized tools and techniques to support application providers in the management of QoS guarantees for real-time interactive applications hosted by IaaS providers. One challenge of course is that IaaS providers do not offer on-demand QoS adaptability in Service Level Agreements (SLA); Amazon EC2 for example only provides the minimum terms for service guarantees with an "Annual Uptime Percentage of at least 99.95% during the Service Year" and a penalty model based on service credits [4].

To achieve greater efficiency in utilization throughout all cloud layers greater interaction and sharing of, for example, QoS measurements will be necessary. A few efforts have been made in this direction to provide QoS guarantees to cloud computing services. Iyer et al. [5] investigate the problem of shared resource contention in virtual machines and propose a model to estimate the shared resources required. One important characteristic of the virtual machines proposed by these authors is their capability to enforce policies to guarantee QoS parameters in SLA. They achieve this task by adopting a class-of-service based cache and memory allocation mechanism in a small number of classes of service. Yu et al. [6] propose a distributed scalable infrastructure for supporting real-time video streams. Also related to cloud computing for real-time systems is the work done by Kim, Beloglazov and Buyya [7], whom analyse power-aware virtual machines architectures with a focus on the optimization of power consumption based on the requirements of a SLA. The QoS requirements are enforced by a Real-time model that uses release time, worst-case execution time, relative deadline, period of service use and finish time as parameters to calculate virtual machine requirements and scheduling tasks. In the area of adaptive cloud systems, Dai et al. [8] propose a self-diagnosis and self-healing tool using a hybrid mechanism composed by a Multivariate Decision Diagrams and Naïve Bayes Classifiers. The goal of their work was to provide a reliable and dependable cloud computing platform. Finally, related to SLA enforcement and provisioning, Hasselmeyer et al. [9] analyse the negotiation of SLA with dynamic policies and prices. In their work they propose an architecture for SLA negotiation that includes an optimizer component that maximizes the offer received by the customer by analyzing knowledge about the resource capabilities of the provider and a dynamic pricing component that computes the price of the SLA according to the provider's available resources, optimization function and resources requested by the customer.

III. Real-Time Interactive Multimedia Applications

Traditionally, 'real time' refers to hard real-time systems, where even a single violation of the desired timing behaviour is not acceptable. However, there is also a wide range of applications that also have stringent timing and performance needs, but for which some deviations in Quality of Service (QoS) are acceptable, provided these are well understood and carefully managed. These are soft real-time applications and include a broad class of interactive and collaborative tools and environments, including concurrent design and visualization in the engineering sector, media production in the creative industries, and multi-user virtual environments in education and gaming. In particular, we focus on interactive soft real time applications where one or more users interact with the application and with each other.

Soft real-time applications are traditionally developed without any real-time methodology or run-time support from the infrastructure on which they run. The result is that either expensive and dedicated hardware has to be purchased to ensure good interactivity levels and performance, or that general-purpose resources are used as a compromise (e.g. commodity operating systems and Internet networking) with no way to guarantee or control the behaviour of the application as a result. For such applications PaaS needs to support techniques for modelling, predicting, provisioning and monitoring resource and QoS requirements commitments and applying such techniques in a general way so they can be exploited in different application domains.

The IRMOS PaaS architecture is driven by real-time multimedia applications from business sectors including post production, virtual augmented reality and e-Learning, each providing QoS requirements in respect to the on-demand provisioning of virtualized infrastructure resources. An example from the post production scenario is shown in figure 1 based on the Digital Film Technologies Bones Digital Dailies production system [10]. In the scenario, collaborative and distributed colour correction is performed as part of film post-production. A post-production house is contracted to perform colour correction to some film shots that will be selected by a film director during his review of the digital dailies of a film currently under production. The number of shots needing colour correction cannot be determined in advance as this depends on decisions made by the director. The director estimates that colour correction will be applied to approximately 30 +/- 10 minutes of footage. The colour correction and review activities occur concurrently and

consists of: (1) the colourist effects specialist downloads the digitised video from the Bones Service provider who provides storage, processing and networking resources procured from the Cloud, (2) the colourist applies initial correction to the video, by streaming it from the service provider to the post-house so they can determine the correction settings needed, and (3) the colourist and director interactively review the corrections that are applied through real-time stream processing of the video using applications installed at the service provider.

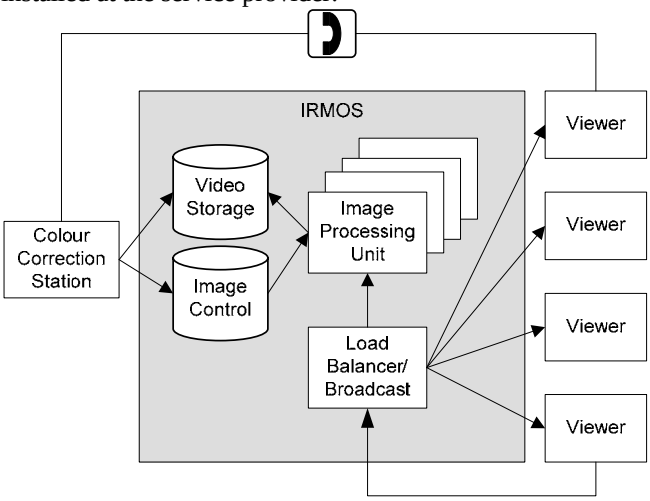

Figure 1. Post-Production Application for Collaborative Colour Correction

The application consists of storage for video and control metadata, a colour correction station, a variable number of image processing units depending upon the data rates and a load/balancer broadcaster responsible for delivery of the stream to the views. In the scenario shown in Figure 1, we use 4 image processing units at 24 frames per second. Each image processing unit queries the image processing control for the current set of colour correction parameters and caches these locally in order to be able to operate as quickly as possible on any frame. The load balancer and broadcaster component is the heart of the system with a clock ticking at the specified frames per second and a local buffer of processed frames. The load balancer instructs each of the four image processing units to process frame one by one. An image processing unit will retrieve an uncompressed frame from the video storage at 2500 Mbps, apply the colour correction according to its local parameter cache, downsize the original 12.8 MB frame to the size required for streaming (approximately 0.07 MB) and apply JPEG compression. The broadcaster part has a separate connection to each connected viewer through which it pushes the "current frame" according to the clock, in this case four streams at 13 Mbps. If the connection to the viewer is too slow then frames will be skipped. The component also has a control channel with each of the viewer components through which it can receive start, stop and seek instructions. During the review process the colourist may be asked to adjust the colour correction for a particular section of video. The colourist will work on this scene using the colour correction workstation in the same way as the initial phase and publish the correction of the parameters to the image processing control which notifies the image processing units of the update. The image processing units then request the updated parameters and update their local caches accordingly.

IV. PAAS ARCHITECTURE FOR REAL-TIME INTERACTIVE MULTIMEDIA APPLICATIONS

A PaaS architecture supporting real time interaction between distributed set of people and resources requires the following key features:

- **Real-Time QoS Specification**: specification language and associated toolkit for the specification of application service components considering both structure and real-time QoS.
- **Event Prediction**: QoS oriented service engineering models for predicting QoS requirements, using temporal and probabilistic profiles of application and resourcing events.
- **Dynamic SLA Negotiation**: SLA negotiation and management services supporting the dynamic negotiation (self-service) of Application-SLAs considering customer requirements and dynamic discovery of resource providers (Technical-SLAs).
- **On-Demand Resource Provisioning**: provisioning services for application service components on virtualised infrastructures through combination of workflow and service-based management wrappers.
- **QoS Event Monitoring**: monitoring services for measuring quality of service at both application and technical levels.

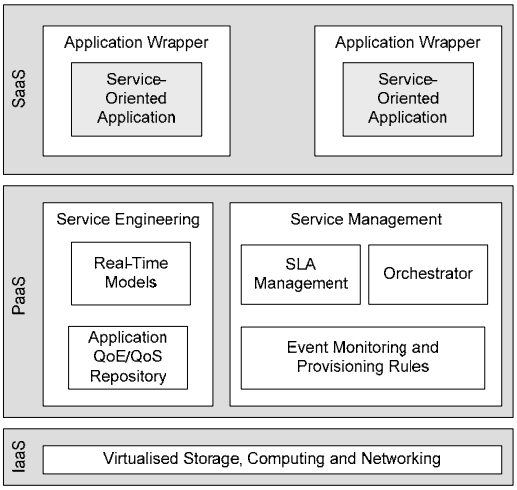

Figure 2. PaaS Architecture

The PaaS architecture is shown in figure 2 and shows the core components and interactions with both SaaS and IaaS. The architecture consists of two main elements, service engineering and service management, which are described in more detailed in the subsequent sections.

A. *QoS oriented Service Engineering*

Quality of Service oriented Service Engineering (QoSSE) supports two important features for real time systems: real-time specification for applications and event prediction. The activities in service engineering are shown in figure 3. Real-time specification is related to how an application is represented to the PaaS architecture and event prediction is associated with the procedures and mechanisms needed to model an application and determine the appropriate infrastructure requirements.

The QoSSE interact with application's developers and with the application itself through a *Development Interface*. It would be very difficult to monotonically describe applications; instead applications are broken into application components (AC) developed by an application component developer. These components are referred as Application Service Component (ASC). In order to use ASC, these need to be described and registered (at the Application QoS/QoE repository in figure 2). Quality of Experience (QoE) allows a service provider to make observations that may differ from the QoS guarantees owing to factors outside of the service provider's control. The consumer may use QoE measurements to validate the QoS reported to it by the service provider, but must recognise that any discrepancy may be due to factors outside of the terms of the SLA (e.g. local network latency). The Application Service Component Description (ASCD) comprises the definition of the input and output interfaces of an ASC as well as the required computing and network resources, which may be depending on the input and output formats actually used as well as timing constraints. ASCD are based in QoS specifications modelled using UML. In particular the UML Profile for Modelling and Analysis of Real-time and Embedded Systems (MARTE) [11] and UML for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms [12] are used.

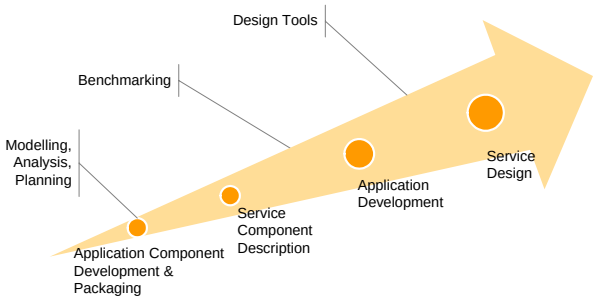

Figure 3. Service Engineering

The QoSSE interfaces with Service Management through a *Performance Estimation Interface* that is used to calculate resource provisioning policies from customer requirements and application constraints (i.e. image resolution, required video streaming parameters, maximum completion time, etc.) in the form of SLA. To calculate the appropriate resources such as processing, memory and network required by the SLA requested, the QoSSE uses a variety of performance evaluation mechanisms such as parameter mapping and statistical modelling tools.

The purpose of the parameter mapping process is to produce mapping rules that can be used to translate the high-level parameters described in the ASCD to QoS requirements. To produce this mapping, benchmarking techniques and Artificial Neural Networks (ANNs) are used. The purpose of benchmarking is to gather a set of data that will come from the test executions of an ASC with different high level parameters on different platforms, characterized by a benchmark index. This index is used to train an ANN that generates mapping rules and provides algebraic functions that allow low level resource parameters (such as CPU cycles, disk usage or network traffic) to be calculated from high level ASC terms (such as fps, resolution, etc) without actual knowledge of the internal source code of the ASC. These functions provide basic knowledge about the behaviour of the ASC and can then be used by subsequent models to calculate completion time probabilities. A detailed description of this process is provided in [13].

The resource requirements generated by parameter mapping do not consider the user's interaction with the application. It would be very difficult to accurately create a model representing the internal application behaviour. However, we can use the knowledge about externally observed behaviour of the application, the behaviour of the execution environment and the pattern of user interactions to build a statistical high level performance model incorporating the behavioural aspects found in real time and interactive systems. This is done by using Finite State Machine modelling techniques [21], specifically we use discrete time stochastic finite state automata and the PRISM model checking tool [14]. PRISM accepts specifications in probabilistic temporal logic that allows us to express probabilistic properties answering questions such as: "The probability to finish task X in Y time", "the probability of a system interruption after Z minutes", etc. The result of these properties is further used in the calculation of the ASC requirements. Details on how we use of this modelling mechanism in a video post-production application scenario can be found in [15].

B. *On-Demand Service Management*

Following the outcomes of the QoSSE tools, IRMOS service management proposes the so-called "Online Process" (as depicted in Figure 4) that includes application concretion (during which the application template is populated with concrete QoS parameter values), discovery & negotiation (referring to IaaS providers), reservation (the IaaS resources are reserved from the PaaS provider), service instantiation (referring to the setup of the virtual service network of the IaaS and the virtual machine units that contain the ASCs), service component configuration (during which the instantiated ASCs are configured according to the application's configuration), execution & monitoring (referring to the execution of the ASCs and monitoring of them as discussed previously in this paper) and cleanup (where the ASCs are stopped and the virtual service network of the IaaS is torn down).

The service management components are responsible for on-demand SLA negotiation, resource reservation, service instantiation, execution and monitoring of Quality of Service for application provisioned at an IaaS provider. In IRMOS the IaaS is provided by ISONI (Intelligent Service Oriented Network Infrastructure) which is described in more detail in [16][17][18]. The service management components are shown in figure 2: (i) SLA Management, (ii) Orchestrator, (iii) Event Monitoring and Provisioning Rules and (iv) Application Wrapper. Each component is now described in more detail in the following sections.

***SLA Management*:** consists of the SLA negotiator, Application-SLA (A-SLA) management and Techincal-SLA (T-SLA) management. The SLA negotiator aims to provide valid SLA offers to the service customer prior to the execution of the services so that the service level requirements can be guaranteed whilst meeting customer's satisfaction. There are two different negotiations involved: (i) A-SLA negotiation between the customer and the SaaS Provider, and (ii) T-SLA negotiation between SaaS provider and IaaS provider. All of these are orchestrated by the SLA negotiator in an automatic manner using resource provisioning policies derived from application models defined during service engineering. The A-SLA manager is responsible for the management of A-SLAs, which includes query, publishing, creation and update SLA templates and mapping commitments to IaaS resources. Furthermore, monitoring information is relayed to the A-SLA manager to detect violations. The T-SLA manager is responsible for the management of T-SLAs which specify resources procured with IaaS providers. T-SLAs are being offered by the IaaS providers as a reply to requests from the PaaS providers. The aforementioned request, namely Virtual Service Network Description (VSND), has been modelled within IRMOS and encompasses information related to the virtual machine units and the network links interconnecting them. This information includes QoS annotations as requests towards the IaaS providers. One of key functionalities of the T-SLA Manager is reporting the SLA violation to the SaaS provider through a notification mechanism that can then be used to trigger events for mitigating management actions. When violation events occurs, the violation information will be sent out to the subscribed party. The implementation of the notification mechanism employs WS-Notification [19].

Orchestrator: The orchestrator is a controller responsible for configuring, starting and stopping the applications. The role of the orchestrator can be two-fold: Firstly, it is responsible for configuring the applications prior to execution, which is achieved by receiving a configuration command from the A-SLA manager. Secondly it executes the workflow based on a real-time enhanced process description based on BPEL4WS. The enactor is aware of temporal constraints for activities within the workflow in addition to current control and dataflow constructs [20].

Event Monitoring and Provisioning Rules: The monitoring service is responsible for collecting the run-time information generated by applications and the IaaS respectively. Functionalities of the monitoring service includes aggregating and storing the acquired data to the historical database (for offline usage - providing input to the QoSSE), and detecting on-the-fly any A-SLA violations (online usage - providing input to the A-SLA manager).

ASC Wrapper: IRMOS involves different type of ASCs (e.g. transcoding, colouring), each of which has its own specific features. In order to decouple the specific ASC from the PaaS services, an ASC Wrapper is developed to 'wrap' the specific ASC binary and provide a unified management interface. The ASC wrapper itself is generic. In order to self-adapt to the different ASCs, it is designed to support three different functionalities: (i) Configuration of the ASCs upon instantiation, (ii) Execution control of the ASC through predefined commands (e.g. start, stop, pause, and resume), and (iii) Acquiring monitoring and state information from the ASCs.

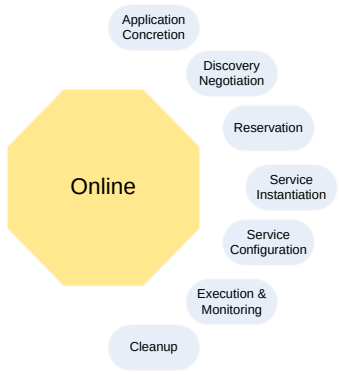

Figure 4. Service Management

The usage sequence of the ASC wrapper is described as follows: (i) upon instantiation of the virtual machine units the ASC wrapper starts automatically (ii) the ASC Wrapper waits to receive a request message from the service management (iii) the ASC wrapper invokes the given command (e.g. configure, control, monitor) and returns back the response (e.g. error code or monitoring data) received as the output of running the application (iv) the wrapper stays inside this 'wait-receive-response' loop to receive subsequent control commands.

V. EVALUATION

We are currently developing the prediction models and provisioning infrastructure needed for our empirical evaluation work. We intend to evaluate application performance models through (1) 'ground truth' benchmarking for specific applications of estimated QoS vs measured QoS and (2) comparison of low level models (e.g. neural networks) with 'close to infrastructure' inputs, such as CPU cycles, to higher level models (e.g. PRISM models) that factor in probabilistic human interactive behaviours. We will (3) simulate business-level eco-systems to evaluate different provisioning strategies, providing evidence regarding optimal strategies that improve customer QoE and provider quality of business (QoBiz) and (4) quantify performance of specific strategies by running instrumented applications on existing cloud infrastructures (e.g. Microsoft Azure).

VI. Conclusions and Future Work

This paper has described a PaaS architecture for provisioning of real-time service-oriented application in clouds. The paper has presented two key aspects of PaaS, namely service engineering and service management, showing how the combination of methods, tools and services can be used to improve the usability, maintainability, efficiency of services targeting clouds with strict QoS constraints.

Our architecture and demonstrators will no doubt identify requirements and opportunities for new PaaS capabilities due to the need to evolve to support increasingly dynamic platforms that can align demand with resource provisioning whilst maintaining guaranteed levels of QoS. PaaS solutions provide an integration layer between SaaS and IaaS, and as such need to mediate concerns and resolve tussles between different vendor viewpoints. This tussle resolution covers all architectural aspects including SLA negotiation, level of sharing of monitoring information, benchmarking abstractions, business and economic aspects, etc.

Key future challenges include addressing the need for dynamic uncertainty management (tracking and decision making) for workflow event probabilities based on event monitoring and runtime adaptation of resource provisioning policies. Here the control loop between service engineering and service management becomes increasingly important and will result in a continued blurring between service design and execution activities.

Acknowledgments

IRMOS is a collaborative research and development project supported by the European Commission under the Seventh Framework Programme FP7/2007-2011, ICT-2007.1.2. IRMOS started in February 2008 and runs for 3 years.

References

[1] IRMOS Project, http://www.irmosproject.eu, 2010

[2] R.Clark, “A Break in the Clouds: Towards a Cloud Definition”, ACM SIGCOMM Computer Communication Review, Volume 39 , Issue 1 (January 2009),pp. 50-55.

[3] http://www.facebook.com/apps/directory.php, 2010

[4] Amazon SLA http://aws.amazon.com/ec2-sla/, 2010

[5] R. Iyer, R. Illikkal, O. Tickoo, L. Zhao, P. Apparao and D. Newell, “VM3: Measuring, modeling and managing VM shared resources”, Computer Networks, Volume 53, Issue 17, Virtualized Data Centers, December 2009, Pages 2873-2887

[6] T. Yu, B. Zhou, Q. Li, R. Liu, W. Wang and C.Cheng, “The design of distributed real-time video analytic system”, Proceedings of the first international workshop on Cloud data management, Cloud-DB 09, Hong Kong, 2009

[7] K. Kim, A. Beloglazov and R. Buyya, “Power-aware provisioning of Cloud resources for real-time services”, In Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, MGC 09, Urbana Champaign, Illinois, 2009

[8] Y. Dai, Y. Xiang and G. Zhang, “Self-healing and Hybrid Diagnosis in Cloud Computing”, Lecture Notes in Computer Science, Volume 5931/2009, Pages 45-56, Springer 2009

[9] P. Hasselmeyer, B. Koller, I. Kotsiopoulos, D. Kuo and M. Parkin, “Negotiating SLAs with Dynamic Pricing Policies”, In Proceedings of the SOC@ Inside’07, 2007

[10] http://www.dft-film.com/software/bones_dailies.php, 2010

[11] S. Gérard, D. Petriu and J. Medina, “MARTE: A New Standard for Modelling and Analysis of Real-Time and Embedded Systems” by, 19th Euromicro Conference on Real-Time Systems (ECRTS 07), Pisa, Italy, July 3rd, 2007.

[12] "UML Profile for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms", Object Management Group, 2004

[13] G. Kousiouris, S. Gogouvitis, D. Kyriazis and T. Varvarigou, Intellegent Mapping of High-Level Application Terms to Resource Level Attributes In Service Oriented Infrastructures, “unpublished”

[14] PRISM web site http://www.prismmodelchecker.org, 2010

[15] M. Addis, Z. Zlatev, B. Mitchell and M. Boniface, “Modelling Interactive Real-time Applications on Service Oriented Infrastructures”, 2009 NEM Summit, ISBN 978-3-00-028953-8

[16] ISONI Whitepaper http://irmosproject.eu/Files/IRMOS_WP6_7_ISONI_White_Paper_ALUD_USTUTT_v2_0.pdf, 2010

[17] M. Kessler, S. Braun, K. Oberle and D. Lamp, “Network Virtualization: Towards a fully virtualized Service Infrastructure”, Proceedings of International Supercomputing Conference, ISC’09, Hamburg

[18] T. Cucinotta, G. Anastasi, and L. Abeni, “Real-Time Virtual Machines”, 19th IEEE Real-Time System Symposium, RTSS 2008

[19] WS-Notification, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn, 2010

[20] S. Gogouvitis, G. Kousiouris, K. Konstanteli, T. Polychniatis, A. Menychtas, D. Kyriazis and T. Varvarigou, “Realtime-enabled Workflow Management in Service Oriented Infrastructures”, Proceedings of AREA2008 workshop of ACM International Conference on Multimedia

[21] S. Balsamo, A. Di Marco, P. Inverardi and M. Simeoni, "Model-Based Performance Prediction in Software Development: A Survey", IEEE Transactions on Software Engineering, Vol. 30, No. 5, May 2004