ASCETiC: Adapting Service lifeCycle towards EfficienT Clouds

Ana Juan[1], David García[1], Eleni Agiatzidou[2], Francesc Lordan[3], Jorge Ejarque[3], Raül Sirvent[3], Rosa M. Badia[3], Jordi Guitart[3], David Ortiz[3], Mario Macías[3], Jean-Christophe Deprez[4], Christophe Ponsard[4], Christian Temporale[5], Pasquale Panuccio[5], Davide Sommacampagna[5], Lorenzo Blasi[5], Karim Djemame[6], Django Armstrong[6] and Michael Kammer[7]

[1]Atos Spain, SA, Avinguda Diagonal 200, 08018 Barcelona, Spain
[2]Athens University of Economics and Business, 76, Patission Street, GR10434 Athens, Greece
[3]Barcelona Supercomputing Center, Jordi Girona 31, 08034 Barcelona, Spain
[4]Centre d'Excellence en Technologies de l'Information et de la Communication, Rue des Frères Wright, 29/3, B-6041 Charleroi, Belgique
[5]Hewlett Packard Italiana SRL, Via Giuseppe Di Vittorio, 9, Cernusco sul Naviglio, MI 20063 Italy
[6]University of Leeds, Leeds LS2 9JT, U.K.
[7]Technische Universität Berlin, Straße des 17. Juni 135, D-10623 Berlin, Germany
{ana.juanf, david.garciap}@atos.net, agiatzidou@aueb.gr
{frances.lordan, jorge.ejarque, raul.sirvent, rosa.badia, jordi.guitart, david.ortiz, mario.macias}@bsc.es
{jean-christophe.deprez, christophe.ponsard}@cetic.be
{christian.temporale, pasquale.panuccio, davide.sommacampagna, lorenzo.blasi}@hp.com
{K.Djmame, scsdja}@leeds.ac.uk, michael.kammer@tu-berlin.de

Abstract Reducing energy consumption is increasingly gaining attention in the area of Cloud computing, as means to reduce costs and improve corporate sustainability image. ASCETiC is focused on providing novel methods and tools to support software developers to optimise energy efficiency and minimise the carbon footprint resulting from developing, deploying and running software in Clouds. At the same time, quality of service, experience and perception are still taken into account, so energy efficiency will complement them and boost Cloud efficiency at several dimensions. ASCETiC primary focus is to relate software design and energy use, which will depend on the deployment conditions and the correct operation of the software by means of an adaptive environment. This paper presents specific objectives for the project, as well as requirements, business goals and architecture for the resultant Open Source Cloud stack providing energy efficiency at software, platform and infrastructure Cloud layers.

1 The ASCETiC Project

Information and Communication Technology (ICT) constitutes a diverse array of economic, industrial and social activities. Energy efficiency is increasingly important for its future. The increased usage of ICT, together with growing energy costs and the

Ponsard C., Deprez J., Kammer M., Ferrer A., Lordan F., Ejarque J., Agiatzidou E., Ortiz D., Temporale C., Blasi L., GarcÃ■a D., Macias M., Sirvent R., Panuccio P., Sommacampagna D., Djemame K., M. Badia R., Guitart J. and Armstrong D.
ASCETiC - Adapting Service lifeCycle towards EfficienT Clouds.
DOI: 10.5220/0006183400890106
In *European Project Space on Information and Communication Systems (EPS Barcelona 2014)*, pages 89-106
ISBN: 978-989-758-034-5

need to reduce greenhouse gases emissions call for energy-efficient technologies that decrease the overall energy consumption of computation, storage and communications. Forecast for ICT [1] indicate a steady growth of carbon footprint (CO2) of about 4 percent per year, accounting for a total of around 70 percent growth between 2007 and 2020, when the estimated total carbon footprint for the ICT sector will be about 1,100 million tons in 2020.

Enhancing the energy efficiency of ICT is therefore important, not only to reduce power consumption and CO2 emissions, but also to stimulate the development of a large leading-edge market for ICT-enabled energy-efficient technologies that will foster the competitiveness of the industry, improved Return on Investments (ROIs) and result in new business opportunities.

Google, the most-used Internet service, provides a very good example for understanding energy usage and carbon footprint of Cloud services. It is estimated that Google manages over one million servers and processes one billion search request daily. The operation, production and distribution of these servers produce huge amounts of CO2, assessed, depending on the source, of a value between 0.2g [2] and 1g [3] of CO2 per search, amounting to a daily total of one thousand tons of CO2. This is just one example, but it offers an order of magnitude to assess the ecological impact of day-by-day ICT activities.

Traditionally Green IT research and development have focused on energy efficiency for hardware and data centre facilities. Energy efficiency for software has only been marginally considered, mostly in mobile computing, aiming to optimize the duration of devices' batteries. Although it has a direct impact on system's energy consumption; software usually controls how computing is utilized. Covering the full service lifecycle, from application design, development, deployment and operation, it is crucial to determine and optimize the energy usage of the complete system, considering software and hardware as interrelated mechanisms.

Cloud computing presents a model in which IT infrastructure is leased and used according to the need of the enterprise. The benefit of this model is that it converts capital expenditure of an enterprise into operational expenditure. Although building, deploying and operating applications on a Cloud can help to achieve speed, scalability and maintain a flexible infrastructure, it brings about a variety of challenges due to its massive scalability, complexity, as well as dynamic and evolving environments.

ASCETiC is concerned with the topical issue of energy efficient computing, specifically focusing on design, construction, deployment and operation of Cloud services. It argues that research is needed to propose novel methods and develop tools to support software developers in monitoring, minimizing the carbon footprint and optimizing energy efficiency resulting from developing and deploying software in Cloud environments. Such research addresses the need for continued development of infrastructure support for Clouds in order to optimize, monitor and reduce carbon footprint and costs for Cloud providers and end users. The major contribution to the carbon footprint of IT software in general is energy consumed in its operation, thus the primary aim of ASCETiC is to relate software design and energy consumption. Although energy use is of relevance across all software design and implementation, ASCETiC makes specific reference to Cloud-based service operations: the emergence of Cloud computing with its emphasis on shared software components which are likely to be used and reused many times in many different applications makes it

imperative that the software to be developed is as energy efficient as it possibly can be.

Therefore, ASCETiC primary goal is to characterize the factors which affect energy efficiency in software development, deployment and operation. The approach focuses firstly on the identification of the missing functionalities to support energy efficiency across all cloud layers, and secondly on the definition and integration of explicit measures of energy requirements into the design and development process for software to be executed on a Cloud platform. ASCETiC will measure how software systems actually use Cloud resources, with the goal of optimizing consumption of these resources. In this way, the awareness of the amount of energy needed by software will help in learning how to target software optimisation where it provides the greatest energy returns. To do so, all three layers in Cloud computing: Software, Platform and Infrastructure will implement a MAPE (Monitor, Analyse, Plan and Execute) loop. Each layer monitors relevant energy efficiency status information locally and shares this with the other layers, assesses its current energy status and forecasts future energy consumption as needed. Actions can then be decided and executed according to this assessment.

1.1 Vision and Approach

Basically, the ASCETiC approach focuses firstly on the identification of the missing functionalities to support energy efficiency across all Cloud layers, and secondly on the definition and integration of explicit measures of energy and ecological requirements into the design and development process for software which can be executed on a Cloud platform.

ASCETiC's goal is to characterise the factors which affect energy efficiency in software development, deployment and operation. Our main novel contribution is the incorporation of a novel approach that combines energy-awareness related to Cloud environments with the principles of requirements engineering and design modelling for self-adaptive software-intensive systems.

Therefore, the objectives of ASCETiC are:

Objective 1: To extend existing development models for green software design, supporting sustainability at all stages of software development and execution.

Objective 2: To develop and evaluate a framework with identified energy efficiency parameters and metrics for Cloud services.

Objective 3: Develop methods for measuring, analysing and evaluating energy use in software development and execution.

Objective 4: To integrate energy efficiency into service construction, deployment and operation leading to an Energy Efficiency Embedded Service Lifecycle.

2 Requirements Gathering and Elicitation

To achieve the vision of an efficient Cloud operation, the ASCETiC project starts with specifying the business and technical requirements for developing appropriate

methods and tools matching with Industry needs. To analyse business and technical requirements in parallel, two complementary approaches are followed.

For initiating business requirements gathering, a market analysis [4] sets up the stage. Besides, interviews with the two Industry ASCETiC partners, namely ATC and GreenPrefab, as well as other 15 interviews with Industry members beyond the ASCETiC consortium help to derive a set of general business goals to achieve when building the ASCETiC solution.

For eliciting technical requirements, ASCETiC partners initially performed a thorough review of the state of the art [5] which provided some useful insight towards ASCETiC architecture. From the state of the art review and initial architecture reflection, the goal-oriented requirement engineering method KAOS (Knowledge Acquisition in Automated Specification) [7] is performed to hierarchically specify technical goals and finally elicit technical requirements.

2.1 Requirements Analysis

Capturing energy related requirements is challenging because in standard requirements classifications such as the ISO-9126/ISO25030, there is little room for energy requirements except some weak resource-related notion with no reasoning capability about energy requirements and potential interdependencies or even conflicts with other kinds of non-functional requirements like security, availability, time-responsiveness... Actually most of the time energy requirements are not captured as stressed by [6] and there is a need for better elicitation process to capture energy issues such as the available power budget requirements engineers can work with, how to reason about energy requirements across the design space or event at runtime. This can be achieved using a goal-oriented requirements engineering methodology such as KAOS [7]

Explicitly capturing and reasoning about energy requirements is key to have a better understanding of the impact in terms of costs. Currently, the billing of an ICT service is hardly connected to its "real" energy cost and thus lacks transparency [8].
The ASCETIC vision of how to deal with energy requirements together with other types or requirements and how to manage them is divided in three stages:

- *At requirements time:* explicitly dealing with energy goals as described above. Organising them into a reusable pattern library is also useful.
- *At design time*: identifying relevant energy-related feature of the target architecture and possible variability points which can be used to explore the designed space. Practical means such as UML annotation can be used to support this activity and relate them with energy requirements.
- *At run-time:* translating requirements into a Service Level Agreement (SLA) (i.e. possibly “green SLA” for energy requirements) with associated Key Performance Indicator (KPI). A dynamic architecture can dynamically enforce those SLA based on a "Collect-Analyze-Decide-Act" self-adaptability control loop as described in [9].

Important research questions that will be addressed by ASCETiC are the normalisation of energy measurements, the mapping between hardware, VM and software level, the management of KPIs of contributing/conflicting goals as well as the identification of variability points available for (self)-adaptation.

2.2 Stakeholders and Questionnaires Main Findings

To propose a solution appropriate to the Cloud market, ASCETIC collected business requirements by consulting staff and management of various companies covering most pertinent roles identified in the previous section. This process has been performed by conducting 17 interviews with relevant stakeholders that constitute a representative sample of Cloud professionals today, what includes different roles such as Cloud service providers, Cloud users and a Cloud business analyst.

Except for roles from Intermediaries, such as Deployer and Auditor, where the sample of interviewees is minimal, for all other roles, the number of interviewees is adequate for gathering enough information to steer ASCETIC in an appropriate direction. Cloud Customers expectations are of special interest at this stage, as having a clear understanding of their motivations and views will help craft an ASCETIC solution that align with this market needs.

2.2.1 Questionnaires Main Findings

ASCETiC motivation is built on top of EU's Europe 2020 Strategy for smart, sustainable and inclusive growth and of the transition to a resource efficient economy. It was therefore important to get insights about sample interviewees' awareness of this initiative. In this regard, results are positive, 70% (12) have previous knowledge on the initiative, of course at different levels depending on the profile of the interviewee. In research organizations, 100% of the interviewees are aware of Europe 2020 Strategy, 60% in commercial organizations and 50% in public organizations.

Cloud Business Customer Role

90% of interviewees are interested in services consuming less energy. However the decision factor and business driver for Cloud service selection is cost. The capability to assess energy consumption by means of real time monitoring mechanisms and measurements is perceived as an interesting feature, however there has to be a balance among ecological and economical sustainability. 56% of the interviewees would choose a similar eco-service at the same cost, and the average cost reduction to consider it would be 7%.

Only 50% of the interviewees perceive energy consumption as a problem from a customer perspective. When negative responses occurred, emphasis was put on highlighting the fact that it is a cost factor for providers not for consumers of the services. Those ones considering it a problem report that the main drivers for consumption of eco-efficient services are corporate social responsibility, cost and eco-sensitivity.

The majority of the interviewees (nearly 60%) consider that new pricing schemes will charge users based on their actual energy consumption are fair. However, interviewees make an important remark: users have to be able to properly control and monitor both resource usage and energy metrics.

Responses show that cloud customers (65%) will be willing to sacrifice certain QoS parameters such as performance, security or availability but only by their specific selection and for certain services that are not mission critical. For all the rest,

QoS agreed in the SLA have to be of mandatory fulfilment for the provider and performed optimisation to keep agreed service terms.

Penalties and Rewards are not considered of much interest by Cloud Business Customers, less than 20% considers them as a potential motivation. The other motivations that make customers consider a service, in addition to cost, are: use of Green or Brown energy sources (75%) and image (33%), in addition to cost.

Service Business Operator

Operator roles that consider energy consumption an important problem or a very important problem are approximately 57% of the interviewees and only two of the ones that are responsible in their organization of the energy bill don't consider it an important topic. Motivations for energy concerns are clearly cost and to a lower extent corporate social responsibility, emissions and organization image.

Service providers' interviewees (71%) understand that offerings that consider energy consumption similarly to other quality of services aspects could be beneficial for their business.

Service providers interviewees find that the most effective action that relevant authorities could implement to encourage them to propose lower energy consumption or emissions services would be Tax Rebates. Eco-labels are perceived as interesting but clearly to a lesser extent. Interestingly some of the interviewees think that adoption of greener services will happen independently of policies, although they represent less than 10% of the interviewees.

Displaying energy behaviour is perceived in general as positive by 62% of the service providers interviewed however it is clear from the responses that there is not yet a market push in the IT/Cloud sector. Respondents associate this to a reduced target of their customer base and a need for cheaper services.

From the providers' perspective the incentive that is found more interesting to grant customers in order to promote energy responsible behaviour is to provide Eco-labels to customers. The use of flexible eco-aware pricing models as an incentive is perceived of less interest. Based on this, it seems the providers' interest is more aligned with internal cost reduction than in making their customers being part of the process.

3 Business Goals

This section presents the project Business Goals, based on the analysis of the answers to the previous business questionnaire plus an analysis of the current market context.

- *Greener ICT:* The ASCETiC toolset will provide the appropriate tools to support software developers in monitoring the carbon footprint of their applications and in optimising the total energy efficiency from the design, development and deployment of the software in Cloud environments.
- *Ecological and Economical Sustainable Ecosystem:* ASCETiC energy models, profiles/footprints of Cloud platforms and the real-time monitoring mechanisms and measurements have to make possible the creation of new pricing schemes

that will charge users based on their actual consumption and more efficient use of energy in Cloud resources.

- *Cost-effective Corporate social responsibility:* ASCETiC has to support organisations in addressing increasingly important efforts to be more socially responsible corporate citizens while at the same time optimize cost-cutting efforts by being able to assess and enhance Cloud services energy consumption.
- *Transparency for the customer:* The ASCETiC methods and tools to eco-efficiently manage the life cycle of Cloud services from requirements to runtime through construction, deployment and operation have to consider transparency for the user in all decisions that affect final cost for the users. This means consideration of energy metrics for SLA and monitoring, but also visibility in a provider's resources consumption and optimization applied policies.
- *Support for provider's pursuit of Cost and Energy reduction:* ASCETiC tools and methods have to support providers to achieve their ambition of reducing costs in services provision and operation.
- *Support for provider's pursuit of Cost and Energy reduction:* ASCETiC solutions will gain impact focusing on optimisation of energy efficiency on server side computing and server side storage by validation and demonstration through two real-world applications that demonstrate ASCETiC in two contexts: Computing-intensive applications, and Data-intensive applications.
- *Use of de-facto market standards and standards:* The ASCETiC solution should maintain interoperability with and use of existing widely used solutions as part of the entire service lifecycle.

4 Overall Architecture

The aim of the ASCETiC architecture is to support self-adaptation regarding energy and eco-efficiency while at the same time, remain aware of the impact on other quality characteristics of the overall cloud system such as performance. Therefore, the lifecycle of Cloud services from requirements to runtime through construction, deployment, operation, and their adaptive evolution over time is managed. The energy-efficiency of Cloud resources across the entire cloud software stack is also addressed.

This section provides an overview of this architecture. It includes the high-level interactions of all architectural components present in three distinct layers (SaaS, PaaS, and IaaS) and follows the standard Cloud service life cycle model (construction, deployment, operation).

In the SaaS layer a collection of components interact to facilitate the modelling, design and construction of a Cloud application. The components aid in evaluating energy consumption of a Cloud application during its constructions. A number of plug-ins is provided for a frontend Integrated Development Environment (IDE) as a means for developers to interact with components within this layer. Lastly, a number of packaging components are made available to enable provider agnostic deployment of the constructed cloud application, while maintaining energy awareness.

The PaaS layer provides middleware functionality for a Cloud application and facilitates the deployment and operation of the application as a whole. Components

within this layer are responsible for selecting the most energy appropriate provider for a given set of energy requirements and tailoring the application to the selected provider's hardware environment. Application level monitoring is also accommodated for here, in addition to support for Service Level Agreement (SLA) negotiation.

Finally, in the IaaS layer the admission, allocation and management of virtual resource are performed through the orchestration of a number of components. Energy consumption is monitored, estimated and optimized using translated PaaS level metrics. These metrics are gathered via a monitoring infrastructure and a number of software probes.

4.1 SaaS Layer

The main goal of the SaaS Layer is to provide means to a SaaS development team to measure energy consumption data for elements of a SaaS service or application. For instance, from Industry pilot requirements from ATC and GreenPrefab, it is expected that the ASCETIC development environment should help them to obtain energy measurement for selected components, sub-system or feature of their software solution.

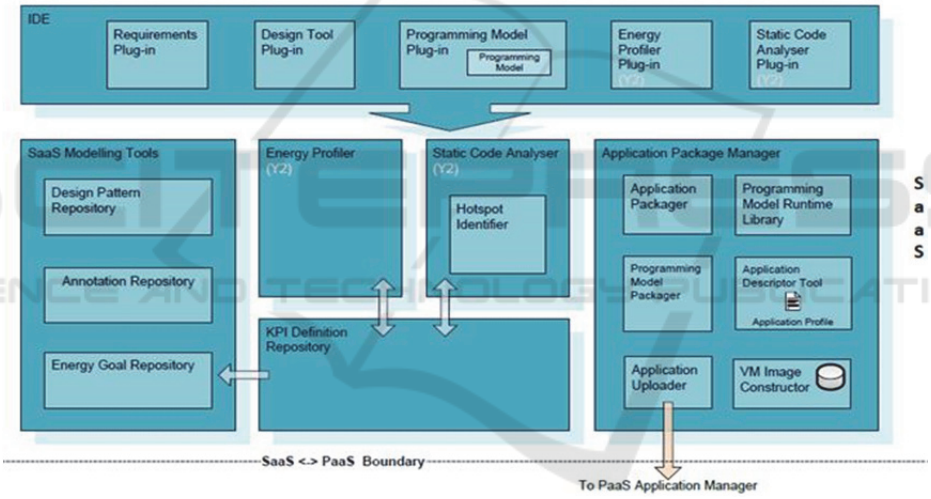

Fig. 1. Architecture overview of SaaS layer.

4.1.1 IDE

The Integrated Development Environment (IDE) of the ASCETiC framework is intended to be the main entry point to the infrastructure for service designers and developers. The IDE integrates the graphical interfaces to the different tools available in the SaaS layer, thus offering a unified and integrated view to users. To achieve such a goal, we have selected Eclipse [10] as the basic tool to integrate all SaaS layer GUIs.

Programming Model
The ASCETiC Programming Model (PM), based in COMPSs [11], provides the service developers with a way to code services (each of its method called Orchestration Element (OE)) composed by pieces of source code, legacy applications executions and external web services (called Core Elements (CEs)). Although these complex services are written in a sequential fashion without APIs, the applications are instrumented so they call the Programming Model Runtime to be executed in parallel.

Programming Model Plug-in
The ASCETiC Programming Model plug-in (PM plug-in) provides a GUI to use the ASCETiC PMl and supporting tools to enable the development, analysis and profiling of an application in order to improve energy efficiency. It helps in creating the service packages, installing these packages in the VM images and creating the Service Manifest where Service Developers may specify any energy and non-functional requirements to achieve a deployment in the Cloud that satisfy their needs.

4.1.2 SaaS Modelling Tools and KPI Definition Repository

Introducing energy-awareness when developing or refactoring application software for the Cloud requires modelling tools for specifying energy and ecological aspects in relation to other quality constraints. These modelling tools must cater to need of various actors involved in the Cloud application development, deployment and evolution process. In relation to energy, such tools must cover key concerns:
At requirements level: express trade-off between performance, cost, energy, and other quality-related requirements such as flexibility in deadline, response time or even cost against energy budget or greenness of energy used. A pattern repository of non-functional requirements and requirement trade-offs relevant to the Cloud application context should also prove useful to development teams.
At architecture level: Support annotation of requirements constraints onto a (possibly retro-engineered) architecture expressed at the right abstraction level. These annotations are connected to probes, probing mechanisms and probe-activation methods. The SaaS Modelling tools can then provide the appropriate information on probes and probing the Application Packager and Service Descriptor so the desired SaaS application measurements can take place during operation.

In iterative software development lifecycles, all requirements may not be known directly, in particular regarding the feasibility to satisfy non-functional requirement trade-offs. Architecture tools must also provide annotation to express the need for collecting monitoring data on non-functional aspects. Knowledge gathered during an iteration can then be used to understand where requirements or trade-off decisions must be relaxed or where development effort must focus to better address non-functional aspects in a fine-grained manner.

4.1.3 Programming Model Runtime

The Programming Model Runtime (PMR) component is in charge of detecting the dependencies among the CE invocations and managing their proper execution in the

remote resources. Every OE is composed by multiple CE invocations that may have some data dependencies. The PMR monitors every data accessed from the OE and the CE to ensure that these dependencies are respected.

The ASCETiC PM will allow the application developer to define multiple implementations for each CE. This new feature enables the CE scheduler to take into account the performance of each implementation and its energy consumption, allowing applying different policies aiming to different objectives. We aim to define a generic interface for scheduling, so multiple policies can be easily developed and the runtime can use them in a plug-in fashion, with different optimisation goals (i.e. execution time, energy consumption…).

Another enhancement is to convert the workers in persistent processes. This will allow the Just in Time (JIT) compiler to optimise the Java byte code of the CD, therefore gaining performance for the applications execution.

4.1.4 Application Packager

The Application Packager is an Eclipse plugin that collects the OVF document submitted by the Application Description Tool, construct the Service Manifest, package the application and submits everything to the VM image constructor.

4.1.5 Programing Model Packager

The Programming Model Packager (PM Packager) component will create the bundles of an ASCETiC Programming Model application. More precisely, it will pack OEs and CEs taking into account their requirements, or any other constraints pointed out by the developer. These application pieces are also compiled, and classes are instrumented in order to intercept OE and CE invocations in the code. It will also generate the Service Descriptor (in OVF format) of the PM application.

4.1.6 Application Description Tool

The Application Descriptor tool provides the users of ASCETiC not using COMPSs a way to describe their Cloud applications to be executed in the ASCETiC platform. The initial selected tool to describe an application it is the Open Virtualization Format (OVF) [12], a widely used standard for the industry to document, in a hypervisor agnostic way, the deployment of one or several virtual machines with it is internal configuration and the relations and dependencies between them.

To help the user writing the OVF file description, a graphical tool is provided, this tool is and Eclipse IDE [13] plugin and relies in a set of Java OVF libraries and the logic to help and guide the user while he/she writes the document.

4.1.7 VM Image Constructor

The Image Creation action requires the Application Package Manager component (precisely, its subcomponent the VM Image Constructor (VMIC)) so that the different

images types can be constructed according to the Service Elements' constraints, and the packages and other service data are installed on these images. Therefore, the VMIC uses the application packages and the service manifest or application descriptor to create VM images that can be deployed in the PaaS layer.

4.1.8 Application Uploader

The Application Uploader component interacts with the Application Manager to register the final VMs ready for deployment. The functionality of this component is very simple, since its mission is to let the final images of an application ready to be deployed by the Application Manager. However, its existence is important because it is a unifying component for the PaaS layer (single contact point). It essentially serves the PM plug-in and the Application Packager in the SaaS layer once images have been completed.

4.2 PaaS Layer

The PaaS layer provides middleware functionality for a Cloud Application and facilitates the deployment and operation of the application as a whole. Components within this layer are responsible for selecting the most energy appropriate provider for a given set of energy requirements and tailoring the application to the selected provider's hardware environment. Application level monitoring is also accommodated for here, in addition to support for SLA negotiation.

This layer is comprised of the Application Manager, the Contextualizer, Application Monitor, the PaaS Energy Modeller, the SLA Manager, the Pricing Modeller and the Provider Registry.

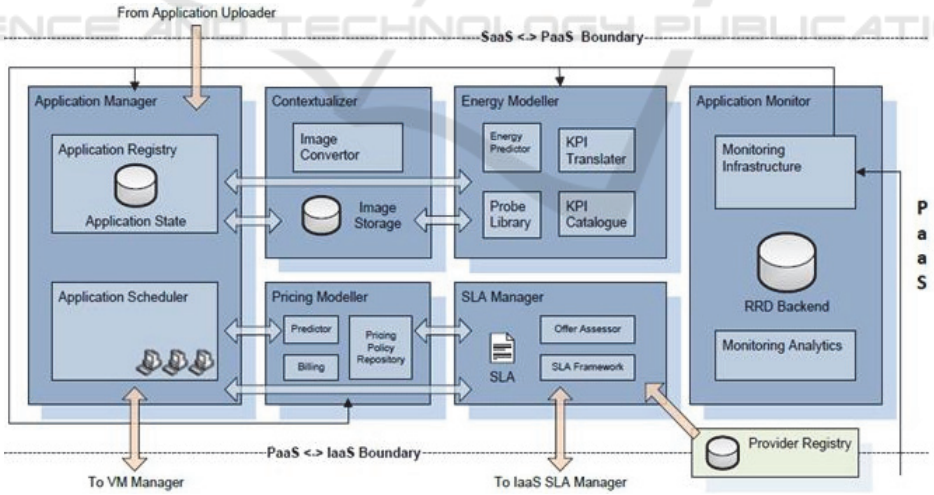

Fig. 2. Architecture overview of PaaS layer.

4.2.1 Application Manager

The Application Manager (AM) component manages the user applications that are described as virtual appliances, formed by a set of VMs that are interconnected among them. Those applications are described in the generated OVF document that contains the different applications requirements. This information is sent to the SLA Manager that makes use of the Provider Registry, Pricing Modeller and Energy Modeller to tell the Application Manager the best facility to deploy the application.

The Application Manager will send the OVF description of the application together with the selected facility to the Contextualizer to prepare the VM images to be deployed. After that it is ready it will deploy the application to the IaaS layer.

While the application it is running, the AM will check its execution accessing the Application Monitoring and take any action if necessary to make sure the requirements of the application are met.

4.2.2 Virtual Machine Contextualizer

The role of the Virtual Machine Contextualizer tools (VMC) is to embed software dependencies of a service into a VM image and configure these dependencies at runtime via an infrastructure agnostic contextualization mechanism [14]. Additionally, the VMC enables the use of energy probes for the gathering of VM level energy performance metrics. It utilizes novel service contextualisation models that require context to be provided at deployment time only, abstracting away the complexities of re-contextualisation at runtime. Additionally re-contextualisation if required is supported. The contextualisation service is invoked by the Application Manager via a single public interface.

The VMC enables one of the key benefits of the ASCETiC project over other competing cloud technologies: simplified energy aware application development. This is achieved by abstracting away the dynamic nature of the Cloud, instead of providing a static environment that increases the ease of service development as resources are moved and migrated [15].

4.2.3 Application Monitor

The Application Monitor collects the KPI-related information that is reported from the application probes that run inside the Virtual Machines. Each application will register its particular KPIs, which may be low-level, such as the amount of used resources (CPU, memory, network...) for a given set of processes, or high-level, such as the rate of service requests per second, the response time, etc. The Application Monitor will store and aggregate them by application, allowing the other components from the PaaS and SaaS layers to retrieve them to check the correct fulfillment of the KPIs and, if required, take reactive actions to ensure it.

In addition, the Application Monitor will perform analytics and visualization of the application metrics to allow the different stakeholders of ASCETiC to refine their models and policies.

4.2.4 Energy Modeller

The goal of the Energy Modeller is to gather and manage energy related information throughout the whole Cloud Service lifecycle and Cloud layers: from requirement level KPIs to COMPS annotations down to PaaS and IaaS level measurements made through the monitoring agents present at those levels.

The PaaS Energy Modeller is responsible for calculating the energy cost of KPIs provided by the PaaS platform. The energy estimation function is based on a model stored inside the Energy Modeller component itself, which will be trained using different workload levels using input vectors provided by the user. The Energy Modeller provides an interface to estimate the energetic cost of a PaaS' KPIs.

4.2.5 PaaS SLA Manager

The component is responsible to manage SLA at PaaS level. The PaaS SLA Manager (SLAM) will negotiate on two sides. On one side it will allow the upper (User / SaaS) layer to negotiate specific guarantees that the PaaS can offer to the application (e.g. if the Platform offers database services it could negotiate the number of offered Third Party Services (TPS)); on the other side the PaaS SLA Manager will negotiate with its IaaS counterparts of different providers to collect the resources needed to fulfil the infrastructure plan (provided by the Application Manager), which will ensure the requested PaaS guarantees. The negotiation with different providers will select the best offer according to predefined criteria.

4.2.6 Pricing Modeller

The Pricing Modeller component is complementing the role of the SLA Manager one. It is responsible for providing billing information and estimations of the total cost that an application will induce when running on top of different IaaS providers. According to the SLA parameters agreed with both, the customer and the IaaS provider through the SLA Manager, the Pricing Modeller is able to calculate or estimate the price or the cost of the application. However, in the calculation and prediction procedure, the component takes into consideration the energy consumed or predicted to be consumed by the application. Also, the Pricing Modeller can provide cost information to the Application Manager in order for the latter to optimise the selection of the appropriate VMs and IaaS provider.

4.2.7 Provider Registry

The Provider Registry keeps a repository of information about the different providers the user applications could be deployed. This information is used by the SLA Manager to study the different features of each provider and to know the information about how to contact it to talk with their respective SLA Manager.

4.3 IaaS Layer

The IaaS layer allocates and manages the virtual resources that are required for the correct operation of the applications while aiming for the fulfillment of the energy-related KPIs (e.g. maximization of energy-efficiency and ecological-efficiency, minimization of energy consumption …) for both the applications and the IaaS layer itself.

In addition, the IaaS layer provides the following information to the PaaS and SaaS layers for their optimal operation: energy and IT-related information from the virtual machines and the services/applications that are running on it, information about the SLAs, the energy consumption of the system and/or pricing information according to the current status of the system.

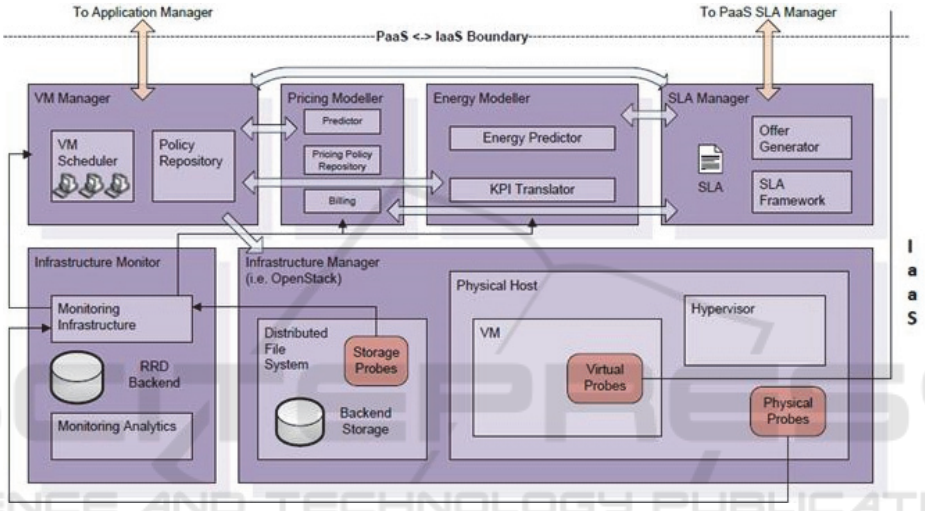

Fig. 3. Architecture overview of IaaS layer.

4.3.1 Virtual Machine Manager

The Virtual Machine Manager (VMM) manages the basic life cycle of the virtual resources (deployment, undeployment, elasticity, and reallocation).

This component continuously performs an optimization process within a changing environment. Considering the input from the PaaS layer, the Infrastructure Monitor, the Price and Energy Modellers, and a set of possible actions over the physical and virtual resources (through the Infrastructure Manager), the VMM must maximize the energy-related KPIs while fulfilling the SLAs that ensure the performance of the applications.

During the deployment of an application, the VMM will decide the physical nodes where the virtual machines will be deployed and/or migrated. Distributing the load across the physical resources would maximize the performance of the applications and minimize the risk of failure. However, physical nodes that are consuming energy with low load during off-peaks are energetically and economically inefficient. Consolidating the maximum number of virtual machines in the minimum number of

physical nodes while switching off the idle nodes would be economically and energetically efficient but it would maximize the probability and failure and minimize the performance of the applications. The VMM must deal with such risks and restrictions to allow a good trade-off between energy/economy and performance.

4.3.2 Energy Modeller

The goal of the Energy Modeller is to gather and manage energy related information throughout the whole Cloud Service lifecycle and Cloud layers. This component's core responsibility is to provide energy usage estimates by presenting the relevant key performance indicators (KPIs) for a virtual machine deployment on the infrastructure provided. This will include cost trade off analysis based on sources such as prior experience, the application profile as defined in the SLA, which is subsequently translated into infrastructure level KPIs, and finally from current up to date monitoring information from the deployment environment.

4.3.3 SLA Manager

The component is responsible to manage SLA negotiation requests at IaaS level. The IaaS SLA Manager cooperates with VM Manager to verify the availability of the required infrastructure configuration and with the IaaS Pricing Modeller to obtain the price for the planned resources with the offered guarantees. Using the information provided by the other components, IaaS SLA Manager is able to generate the specific SLA offer to answer a particular SLA request sent by the PaaS SLA Manager.

4.3.4 Pricing Modeller

Pricing Modeller component of this layer works in the same way as the one of PaaS layer. It is responsible for providing billing information and an estimation of the total cost that a VM will incur to the provider. For the billing and the prediction procedures the total cost or price of the VM running is calculated based on the energy consumed or predicted, the SLA contracted with the customer and a specific pricing policy chosen by the provider. The result of this calculation/prediction will be returned to the SLA Manager, which is the responsible component for cooperating with the upper layer. The VM Manager may also need cost information resulted by this component in order to estimate the optimal VM deployment on top of the infrastructure.

4.3.5 Infrastructure Manager

The Infrastructure Manager (IM) component is responsible for managing the physical resources and to guarantee data security inside the infrastructure. It handles user and service authentication and ensures that only authenticated components are able to observe/modify resources. Applying actions include the inquiring of low-level hardware information like power-consumption, the controlling of the hypervisors and the manipulation of virtualised components like machines, volumes and networks.

Because hundreds of different power meter devices are available on the market and because some of them are not conform to standardised protocols, and additional abstraction layer is needed, especially in a heterogeneous infrastructure. We use a subcomponent to solve this problem. It runs drivers for each hardware meter and provides measurements to the Infrastructure Monitor. Therefore, providers can easily add new metering devices and/or replace one of them with another device model.

The IM is responsible of the deployment of virtual machines (VMs) too. When requested be the VM Manager, it handles the actual deployment on one of the hypervisors in a cluster. The IM has an interpreting role in this case. It answers requests of the Infrastructure Monitor to provide information about available resources, accepts VM images from the VM Manager, reconfigures networking components to guarantee the reachability of a VM and starts the hypervisor on the according cluster node.

4.3.6 Distributed File System

The Distribute File System (DFS) provides an abstraction layer between block devices and storage clients. Clients can access storage objects (e.g. files) the same way as objects on a local file system. We prefer the common well-defined FUSE [16] interface which allows clients to mount the DFS in a local directory. Thus, DFS objects can be accessed like a file. This additional layer comes with a huge advantage: All *NIX programs can make use of the DFS without modification and one DFS module can be exchanged with another one easily.

Our DFS integrates energy-aspects and provides measurements to the Infrastructure Monitor. It shares real-time information like the number of connected clients or the amount of transferred data. Additionally, the DFS timestamps and logs every file access as well as the internal location of an object. That information allows an energy-estimation for future transfers, the identification of appropriate storage servers to save energy for file retrievals and the fair billing of involved DFS clients. The internal object location can help to distinguish between different storage technologies like hard disks, solid state disks, virtual or remote back-ends with sundry energy-metrics. This allows a distinction even if no fine-grained energy-measurements (e.g. in the power cable between hard disk and power supply unit) are available.

4.3.7 Infrastructure Monitor

The Infrastructure Monitor will collect information about resources (CPU, memory, network, etc.) and energy that are being consumed at physical host and virtual machine level and, at the same time, providing historical statistics. This monitoring process should be performed according to two main classifications:

- Performance – among others: physical and virtual nodes characteristics as well as CPU load memory size usage and CPU Frequency.
- Energy – among others: CPU, RAM, Local Disk, SAM and Node Heat Dissipation, Watts per period, as well as % Green Energy is provided to the Cloud infrastructure.

A key factor of the monitoring component for a successful adoption is being able to monitor VMs without forcing the application owner to configure their images with additional tools to establish the communication with the central monitor. Monitoring should be able to pull the information from the VMs in a non-intrusive way.
The solution developed in ASCETiC is based on Zabbix Monitoring tool [17].

5 Next Steps

The ASCETiC architecture is – at time of writing - being implemented on a Cloud computing testbed. The evaluation plan consists in the consideration of two industrial use cases that will be used to exploit and illustrate the ASCETiC architecture, and the possible energy gains it will provide to the applications it operates.

The aim of the first use case will be to adapt an existing computationally intensive software and make it suitable for an energy aware Cloud environment. The Green Prefab (GPF) solution is a product lifecycle management system for the building industry. GPF facilitates the design and construction of a new generation of prefabricated buildings by means of collaborative software and industrial production.

The aim of the second use case will be to adapt existing data intensive software called NewsAsset and make it suitable for an energy aware Cloud environment. NewsAsset has been built to assist news agencies in creating, managing and distributing breaking news quickly and efficiently to various customers through a variety of delivery methods. Its core functionality includes: newsgathering, editing, archiving, managing the production process and delivering services through multiple channels.

To be able to achieve this evaluation, understanding how these applications work from an architectural perspective is the first natural step, so that they can be augmented to run on the ASCETiC architecture. Thus, the next step will include re-engineering and "cloudification" process of the existing use case applications, in addition to the Verification and Validation tasks concerning the energy metrics that will be used to gauge energy efficiency improvements.

The ASCETiC initial tools version focuses on delivering energy awareness in all system components. Monitoring and metrics information are measured at IaaS level and propagated through the various layers of the Cloud stack (PaaS, SaaS) considering static energy profiles. Further versions of the ASCETiC tools will focus on:

- *Intra-Layer Cloud Stack Adaptation:* This iteration relies on previous phase results, adding the capabilities required in order to achieve dynamic energy management per each of the Cloud layers. Adaptation with regard to energy efficiency focuses on an intra-layer approach.
- *Inter-Layer Cloud Stack Adaptation:* This iteration builds on top of intra-layer Energy Efficiency, in order to achieve steering information and decisions among Cloud layers for triggering other layers to adapt their energy mode. The ambition is to enable the entire Cloud stack to actively adapt to changing situations.

Advances on these at the level of new updates to requirements, architectures and even software components will be constantly updated through the ASCETiC website [18].

References

1. Malmodin, J., Bengmark, P., Lundén, D. 2013, The future carbon footprint of the ICT and E&M sectors, Conference paper and presentation at ICT for Sustainability, (ETH Zurich, Switzerland, February 14-16, 2013).
2. Google green, Global perspective, http://www.google.com/green/the-big-picture.html.
3. Columbia University libraries/Information Services, http://journals.cdrs.columbia.edu/consilience/index.php/consilience/article/view/141/57.
4. ASCETiC Market Analysis, http://www.ascetic-project.eu/content/market-analysis.
5. ASCETiC State of the Art, http://www.ascetic-project.eu/content/state-art.
6. Sebastian Götz, Claas Wilke, Sebastian Cech, Uwe Assmann, Runtime Variability Management for Energy-efficient Software by Contract Negotiation. In proceeding of: Proceedings of the 6th International Workshop Models@run.time (MRT 2011).
7. Axel van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley 2009.
8. Lorenz M. Hilty and Wolfgang Lohmann, The Five Most Neglected Issues in "Green IT", Green ICT: Trends and Challenges, CEPIS UP GRADE, Vol. XII, No. 4, October 2011.
9. Betty H. Cheng et als. 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In Software Engineering for Self-Adaptive Systems, Betty H. Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, and Jeff Magee (Eds.). Lecture Notes In Computer Science, Vol. 5525. Springer-Verlag, Berlin, Heidelberg 1-26.
10. "Eclipse Open Source IDE," [Online]. Available: http://www.eclipse.org. [Accessed 16-5-2014].
11. Francesc Lordan, Enric Tejedor, Jorge Ejarque, Roger Rafanell, Javier Álvarez, Fabrizio Marozzo, Daniele Lezzi, Raül Sirvent, Domenico Talia, Rosa M. Badia, "ServiceSs: An Interoperable Programming Framework for the Cloud", Journal of Grid Computing, Springer Netherlands, pp. 1-25, September 2013.
12. Open Virtualization Format, OVF, http://www.dmtf.org/standards/ovf.
13. Eclipse IDE, http://www.eclipse.org/.
14. Towards a Contextualization Solution for Cloud Platform Services. D. Armstrong, K. Djemame, S. Nair, J. Tordsson and W. Ziegler. Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'2011), Athens, Greece, December 2011.
15. Runtime Virtual Machine Recontextualization for Clouds. D. Armstrong, D. Espling, J. Tordsson, K. Djemame, and E. Elmroth. Proceedings of the 7th Workshop on Virtualization in High-Performance Cloud Computing (VHPC'12), Rhodes Island, Greece, August 2012.
16. FUSE, http://fuse.sourceforge.net/.
17. Zabbix Monitoring tool, http://www.zabbix.org.
18. ASCETiC project website, www.ascetic-project.eu.