Open Environment for Collaborative Cloud Ecosystems

Oleksiy Khriyenko
Industrial Ontologies Group, MIT Department
University of Jyväskylä, P.O. Box 35(Agora)
Jyväskylä, Finland
oleksiy.khriyenko@jyu.fi

Michael Cochez
Industrial Ontologies Group, MIT Department
University of Jyväskylä, P.O. Box 35(Agora)
Jyväskylä, Finland
michael.s.l.cochez@jyu.fi

Abstract — **Cloud computing can be defined as accessing and utilizing third party software, services and resources and paying as per usage. It facilitates scalability and virtualized resources over the Internet as a service; providing cost effective and scalable solution to customers. There are two emerging methodologies for constructing infrastructure: "Cloudcenters" and "Infrastructure Web Services". Cloudcenters can be regarded as a virtualized data center. Infrastructure Web Services are more analogous to Service-Oriented-Architectures (SOA), require significant programming skills and are much more comfortable for software developers. It is a robust ecosystem of services which you can use in order to build your application, getting the traditional benefits of Cloud Computing such as self-service, pay-as-you-go, and massive scalability. Unfortunately, talking about openness and interoperability in cloud computing, cloud providers still operate very much in their own silos and private-cloud APIs drift further and further apart. Most data center vendors do not offer users complete vertically integrated cloud stacks. However, they are often providing solutions which imply a strong vendor lock-in. A lot of activities are currently aimed at the development of various Cloud computing environments and software engineering practices for the management of distributed applications, services and other resources. We are thinking about a future vision of a network of clouds. It should be an open market for components (applications, services, data sources, etc.) and composed ecosystem infrastructure services that facilitate appropriate collaboration for personalized needs. In this paper we would like to slightly modify the original cloud stack towards the development of an open environment for task-oriented personalized cloud ecosystems and apply a resource integration platform for this ecosystem elaboration.**

Keywords-collaborative clouds; cloud interoperability; component-based ecosystem infrastructure; semantic integration

I. INTRODUCTION

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and system software in the data centers which provide these services. 'Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry by making software as a service even more attractive and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it.' [1].

Clouds have emerged as a computing infrastructure that enables rapid delivery of computing resources as a utility in a dynamically scalable and virtualized manner. The advantages of cloud computing over traditional computing include: agility, lower entry cost, device independence, location independence, and scalability. There are two emerging methodologies for constructing infrastructure: "Cloudcenters" and "Infrastructure Web Services". Cloudcenters provide the same kinds of tools that data center and server operators are already accustomed to, but with all the advantages of cloud (i.e., self-service, pay-as-you-go and scalability). Instead of creating completely new paradigms, cloudcenters are a methodology by which you, the customer, can have a virtual data center hosted in the "sky". It allows the use of the same tools, paradigms and standards that are deployed in an industry standard data center today. Cloudcenters provide a direct equivalent to traditional data centers including all of the regular components you expect such as hardware firewalls, hardware load balancers, network storage, virtualized servers, dedicated networks, and the option for physical servers for workloads that should not be virtualized. Thus they are usually more desirable for IT staff, systems operators, and other data center specialists. Infrastructure Web Services on the other hand are more analogous to Service Oriented Architecture (SOA), require significant programming skills, and are much more comfortable for software developers. In this case, the infrastructure provides a number of different services (Object-based file storage, Servers on demand, Distributed database functionality, Content distribution, Messaging & queuing, Payment processing, etc.) that can be consumed individually or together to facilitate different kinds of applications. This is a robust ecosystem of services which you can use in order to build your application.

For all the talk about openness and interoperability in cloud computing, both public-cloud and private-cloud providers still operate very much in their own silos [2]. The cloud ecosystem is challenged by the fact that cloud service providers provide their own ways on how users or cloud applications interact with their cloud, resulting in vendor lock-in, non-portability and inability to use the cloud services provided by multiple vendors. This often includes the inability to use an organization's own existing data center resources seamlessly with the offered infrastructure. Cloud computing is gaining popularity and IT giants such as Google, Amazon, Microsoft and IBM have started their cloud computing infrastructure. All of them are doing wonderful things — but they are doing so largely within their own environments. 'And while (most) data center vendors don't offer users complete vertically integrated cloud stacks, they are more than happy to lock users into their product lines as much as possible and form strong partnerships in areas they don't play.'[2]. Golden [3] states that current cloud implementations do not allow enterprise applications to be migrated conveniently; imply legal, regulatory, and business risks; are difficult to maintain ; lack service level agreements and do often not give a cost advantage.

Nowadays, activities are mainly aimed at the development of various Cloud computing environments and software engineering practices for management of distributed applications, services and other resources. However, development is still focused on enterprise level clouds, which may result in the creation of architectures with the drawback of heterogeneity, non-interoperability of components, and inability of the systems to be reconfigurable on demand. Effort is already done in order to make providers' offers interchangeable. One such example is the Open Virtualization Format (OVF). 'The OVF specification is a hypervisor-neutral, efficient, extensible, and open specification for the packaging and distribution of virtual appliances composed of one or more VMs. It aims to facilitate the automated, secure management not only of virtual machines but the appliance as a functional unit.' [4]. The same source states however that 'For the OVF format to succeed it must be developed and endorsed by ISVs, virtual appliance vendors, operating system vendors, as well as virtual platform vendors, and must be developed within a standards-based framework.' This requirement might show to be to strong in reality.

We think that it is time to start thinking about a future vision of a network of clouds. It should be an open market for components (applications, services, data sources, etc.) and composed ecosystem infrastructure services that facilitate appropriate collaboration for personalized needs. Such ecosystem-based environment allows the collection and management of applications and the composition of mash-ups based on them. The applications used to compose mash-ups can be found from the users own private pool of components and services or from the open marketplace provided by different cloud service providers. Furthermore, the ecosystem-based environment allows enterprises and individuals to choose what kind of ecosystem infrastructure services to utilize for the service collaboration and personalized user experience. Such architecture allows us to create personalized abstract clouds. An abstract cloud is a description of infrastructure, platforms and software which does not have to mention all concrete components. These concrete components can later on be selected, even on the fly, by the user of the abstract cloud. Abstract clouds can be made available through the open marketplace to be used as application oriented infrastructure or as a sub-cloud for own personalized infrastructure cloud composition. In this paper we would like to slightly modify the original cloud stack towards the development of an open environment for task-oriented personalized cloud ecosystems. We will apply a resource integration platform for this ecosystem elaboration.

II. SMART RESOURCE INTEROPERABILITY

A. *Technologies Towards Intelligent Interoperability*

With the presence of numerous vendors, the need for interoperability between clouds emerges. The goal is to make complex and developed business applications in the cloud interoperable. To achieve the vision of ubiquitous knowledge, the next generation of integration systems might need different technologies as the ones currently used. Technologies such as Semantic Web [5][6], Web Services [7][8], Agent Technologies [9], and Mobility[10]. Semantic technologies are viewed today as a key technology to resolve the problems of interoperability and integration within the heterogeneous world of ubiquitously interconnected objects and systems. Still, aspects of proactivity of these resources are quite in demand nowadays and should be considered more comprehensively.

In recent years, the complexity of computing environments has grown beyond the limits of human system administrators' management capabilities. With the advent of service-oriented computing (SOC), computing environments have become open and distributed, and components are no longer under a single organization's control. Moreover, the typical enterprise computing environment is a heterogeneous, irregular, multivendor pastiche which is difficult to configure, maintain, and trouble-shoot. Autonomic computing systems are expected to free system administrators to focus on higher-level goals [11]. Self-configuration (systems configuring themselves automatically when computing resources are added or removed), self-healing (discovering when, where and why systems are ailing and performing the appropriate self-repair and fault-correction operations), self-optimization (monitoring and controlling resources to ensure optimal functioning with respect to defined requirements, as well as optimizing performance and efficiency by reconfiguring themselves) can be performed by autonomic computing systems without human intervention. Autonomic computing systems can perform these functions at both the infrastructure and application levels. As such, autonomic computing systems strongly resemble multi-agent systems (MAS). MAS, in turn, interact with services, as designed and developed within SOC. When it comes to developing complex, distributed software based systems; the agent based approach is advocated in Jennings [12]. The vision of autonomic computing emphasizes that the run-time self-manageability of a complex system requires its components to be, to a certain degree autonomous themselves. From the implementation point of view, agents are the next step in the evolution of software engineering approaches and programming languages, a step following the trend towards increasing degrees of localization and encapsulation in the basic building blocks of programming models [13].

Developing and maintaining large-scale, distributed applications is a complex task. Middleware has traditionally been used to simplify application development by hiding low-level details and by offering generic services that can be reused and configured by application developers. However, middleware technology has not kept up with the growing demands that emerge in the digital society: the scale of distributed applications is rapidly increasing, the range of users that compose and configure applications has expanded significantly, and the increased scope of distributed applications has also resulted in more advanced application composition scenarios.

B. *UBIWARE Platform: Integration Infrastructure for Heterogeneous Distributed Components*

As a basis for our research towards open environment for personalized task/domain oriented cloud ecosystems we use the UBIWARE platform [14]. This platform follows the GUN vision described in [15]. The UBIWARE platform is a development framework for creating multi-agent systems. It is built on top of the Java Agent Development Framework JADE [16], which is a Java implementation of IEEE FIPA specifications. The name of the platform comes from the name of the research project, in which it was developed. The UBIWARE project introduced a new paradigm in software engineering and elaborated an approach towards creation of semantically enhanced agent-based integration middleware that makes heterogeneous resources proactive, goal-driven and able to interoperate with each other in collaborative environment [17]. In this project, a multi-agent system was seen, first of all, as a middleware providing interoperability of heterogeneous resources and making them proactive and in a way smart.

The core of the platform gives every resource a possibility to be smart by connecting a software agent to it. This agent enables the component to proactively sense, monitor and control its own state and communicate with other components which are also represented by agents in the system. Furthermore, the component can compose and utilize internal and external experiences and functionality for self-diagnostics and self-maintenance. UBIWARE enables the resources to automatically discover each other and to configure a system with complex functionality based on the atomic functionalities of the resources. It ensures a predictable and systematic operation of the components and the system as a whole by enforcing that the smart resources act as prescribed by their organizational roles and by maintaining the “global” ontological understanding among the resources [18]. The main goal of the platform is to provide interoperability between heterogeneous resources (applications and systems in our case) through semantic adaptation and the proactive agent assigned to each of the resources. All communication, resource discovery and use of resources (e.g., application and systems) are performed trough its corresponding agent. The platform has inter-platform communication mechanisms and allows integration, orchestration and choreography of resources registered and located on different platforms. UBIWARE is not an application like an operating system, word processing software or Internet browser. It is a set of tools that helps people develop software. With respect to cloud-based integration environment interoperability, we consider the UBIWARE platform as a tool that enables automatic discovery, orchestration, choreography, invocation and execution of different Business Intelligence services.

III. Open Environment for Collaborative Cloud Ecosystem

A. *Cloud Stack for Collaborative Cloud Ecosystem*

Most of the current cloud implementations are built on top of data centers. According to NIST [19] cloud computing incorporates Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), and provide these services as utilities. Data centers are a foundation of cloud computing which provides the hardware clouds run on. IaaS is built on top of the data centers and virtualizes the computing power, storage and network connectivity of the data centers, and offers them as provisioned services to consumers. In other words, consumers have the possibility to configure a virtual computer, where he can select a configuration of CPU, memory and storage that is suitable for the intended application. The whole cloud infrastructure (i.e., servers, routers, hardware based load-balancing, firewalls, storage, and other network equipment) are provided by the IaaS provider. The customer buys these resources as a service as needed. Examples of this layer include the Amazon EC2 service [20] and Microsoft’s Windows Azure platform [21]. PaaS provides a development platform with a set of services to assist application design, development, testing, deployment and monitoring, hosted on the cloud. It is sometimes referred to as cloudware. Google App Engine, Microsoft Azure, Amazon Map Reduce/Simple Storage Service, etc are among examples of services residing in this layer. In SaaS, Software is presented to the end users as services on demand, usually in a web browser. It saves the users from the troubles of the software deployment and maintenance. The software is often shared by multiple tenants, automatically updated from the cloud, and no additional license needs to be purchased. Because of its service characteristics, SaaS can often be integrated easily with other mashup applications. One example of SaaS is Google Maps and its mashups across the Internet. However, the separation in IaaS, Paas and Saas is mainly a service model. Components and features of one layer can in practice also be implemented on another layer and the upper layer does not have to be built on top of its immediate lower layer.

In the cloud computing environment, everything can be implemented and treated as a service. Software development "in the cloud" has been one of the really interesting developments to come out of the cloud computing market so far. Regarding PaaS, more and more cloud providers enhance their platforms with specific services that simplify application development for their customers and, in such a way, bind the customers to their platforms. There are services like payment systems, information search systems, GEO-systems, specific data bases, etc. One example of this kind of services is the datasets provided by Amazon [22]. Together with private specific services from cloud providers there are quite many freely open sources and commercial services provided by third parties. To generalize the concept of this kind of services, and take into account that users of such services are not human, but other applications and services, we name them as a SaaS for Software (SaaS4S) or SaaS for SaaS (SaaS4SaaS).

In the proposed solution, a service (be it infrastructure, platform or software) should be registered (connected through adapter) to the integration environment UBIWARE and be semantically annotated according to common ontology in order to enable the environment to discover and orchestrate them. Any service can have access restrictions. This gives an opportunity to use own capabilities together with, or instead of, others when security and privacy of processed data is crucial. Using the UBIWARE platform as a tool for application and service integration, we may create an open environment for the components across different clouds.

Figure 1 shows the proposed extended cloud stack which allows us to organize collaboration between services and applications located in different clouds. As in the original cloud stack, there are Application Development Tools that users of the PaaS layer use to develop and run their applications.

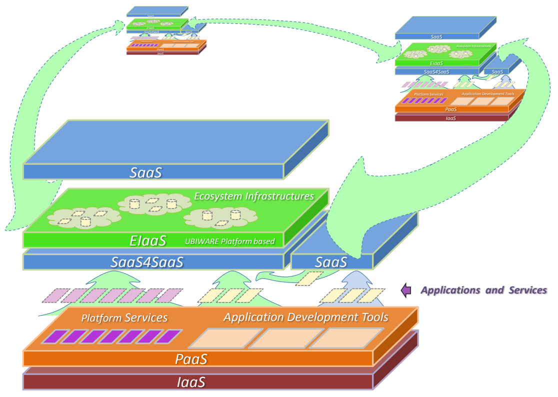
SaaS
Ecosystem Infrastructures
EIaaS
UBIWARE Platform based
SaaS4SaaS
SaaS
Applications and Services
Platform Services
Application Development Tools
PaaS
IaaS

Figure 1. Cloud Stack for Collaborative Cloud Ecosystem.

There are services and functionalities that many of cloud providers supply with their platforms to facilitate users application development. Using the UBIWARE platform as one of the applications run on top of PaaS layer we may:

- transform applications that are presented for humans on the SaaS layer to services available for other software (SaaS4SaaS);
- support the users of the PaaS layer to develop and register applications directly for SaaS4SaaS layer;
- make specific platform services available for use on the SaaS4SaaS layer.

The UBIWARE platform allows semantic adaptation of different data sources and makes them accessible as services for other services and applications that operate through the platform. With correspondent tools (provided by the UBIWARE platform) users may create and define task and domain specific Personalized Ecosystem Infrastructures (PEIs) as compositions of services (addressed by ecosystem Infrastructure Modules) and data sources. They can then use them as services on demand – Ecosystem Infrastructure as a Service (EIaaS). Thus, on the EIaaS layer, the UBIWARE platform provides a possibility to create new services on top of cross-cloud semantic orchestration and choreography of distributed components.

B. Personalized Context-Aware and Self-Configurable Cloud Ecosystem

A component-based approach for the Cloud Ecosystem development provides us a flexible way to elaborate an ecosystem through the composition of different (heterogeneous) modules on the level of Ecosystem Infrastructure and on the level of Application composition (Figure. 2). We utilize the same approach of component-based system development on both levels and provide an interoperability of heterogeneous components (modules) developed by various providers in an open collaborative environment. As a foundation for a collaborative ecosystem environment, we consider a network of platforms that provide cross-platform communication, interoperability of heterogeneous components and a toolbox for their composition. The UBIWARE platform is developed as a smart semantic middleware for ubiquitous computing and is based on integration of several technologies: semantic web, distributed artificial intelligence, agent technologies, ubiquitous computing, SOA, Web X.0 and related concepts. We regard this platform as our basis and intend to extend its functionality towards the needs of this Cloud Ecosystem elaboration.

The Core Ecosystem Engine is an engine which provides a mechanism for a component-based Ecosystem Infrastructure composition. On the Ecosystem Infrastructure development level we have a pool of components – Ecosystem Infrastructure Modules (EIMs). These EIMs can be used for Personalized Ecosystem Infrastructure (PEI) creation where only relevant EIMs are composed. With respect to openness of our collaborative environment, such PEI can itself be published to the public zone and be used as sub-PEI in other personalized ecosystem infrastructures.

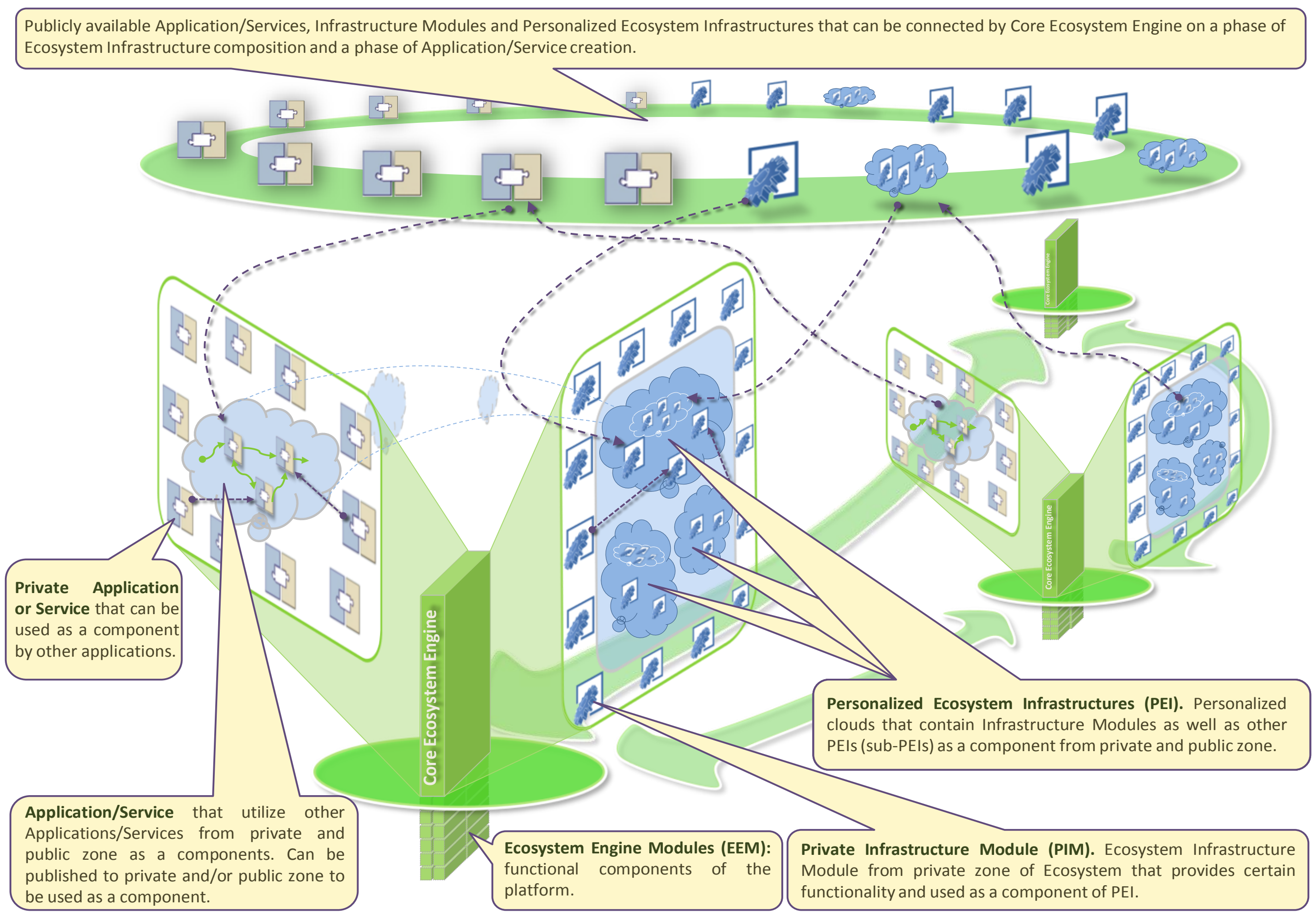
Publicly available Application/Services, Infrastructure Modules and Personalized Ecosystem Infrastructures that can be connected by Core Ecosystem Engine on a phase of Ecosystem Infrastructure composition and a phase of Application/Service creation.
Private Application or Service that can be used as a component by other applications.
Application/Service that utilize other Applications/Services from private and public zone as a components. Can be published to private and/or public zone to be used as a component.
Core Ecosystem Engine
Ecosystem Engine Modules (EEM): functional components of the platform.
Private Infrastructure Module (PIM). Ecosystem Infrastructure Module from private zone of Ecosystem that provides certain functionality and used as a component of PEI.
Personalized Ecosystem Infrastructures (PEI). Personalized clouds that contain Infrastructure Modules as well as other PEIs (sub-PEIs) as a component from private and public zone.

Figure 2. Open environment for collaborative component-based Ecosystems.

In the same way, the Core Ecosystem Engine of the platform provides a mechanism for component-based service composition on top of the selected Personalized Ecosystem Infrastructure. The user may have a private zone containing the set of available applications and services on the platform. At the same time, the platform allows the connection of publicly available services, published to the public zone of the environment. Semantic policy-based control of the platform brings security aspects to the system. This policy based control allows users to distinguish between public and private components and guarantees protection of sensitive data.

In order to make a valuable step towards intelligent services, we should not to limit ourselves to the creation of specific services. We have to think about more flexible solutions that allow us to create new services through orchestration of reusable collaborative intelligence and about a supportive integration environment that gives us the possibility to create new context-aware services through the integration of various data sources and intelligent capabilities with a flexible semantic process. To increase the flexibility and reliability of the applications and the services created on top of Ecosystem Infrastructures, we consider a semantic definition of Abstract Infrastructure. According to the semantic web vision, not only programs and data are distinguishable, but also components of more complex systems are considered as separate modules. These components may be replaced by components which are semantically similar and more suitable in the current context.

Applying a semantic web approach for the Ecosystem Infrastructure creation, the user may define a so called Abstract cloud, which will be on-the-fly transformed into concrete appropriate Infrastructure based on the available components from different Clouds depending on the correspondent context. Providing interoperability of heterogeneous components, Ecosystems should be flexible and at certain level intelligent. Utilizing the semantic web approach, the UBIWARE platform makes the Ecosystem proactive and able to configure itself on-the-fly depending on context and user needs. The platform provides a possibility for the user to define a process with preferences and constrains and executes it as an on-the-fly orchestration of available capabilities and available data, based on their semantic descriptions, through semantic matching and discovery mechanisms (Figure 3).

Although we have a complex network of heterogeneous services, applications and data sources distributed among different clouds, users of EIaaS layer see the Ecosystem as one common entity accessible through the common UBIWARE interface. Figure 4 shows us the general structure of the Ecosystem. The UBIWARE platform provides corresponding tools for the Ecosystem Users (the providers and users of personalized Ecosystem Infrastructures) and transparency for collaboration of distributed components. Through interoperability between other UBIWARE platforms this automatically organizes an open market place of publicly available ecosystem infrastructures and services keeping the possibility of private zones for sensitive information.

C. Case Scenario: Creation of Personalized Ecosystem Infrastructure and Launching Self-configurable Application as a Set of Composed Services

This is a joint use case with two players that shows nested utilization of the ecosystem platform on two layers. It can also be regarded as two separate scenarios.

Player 1 is an ecosystem infrastructure provider who has a set of own components - infrastructure modules. The player would like to create a Personalized Ecosystem Infrastructure (PEI) as a set of (a) own modules, (b) some publicly available modules shared in the open marketplace of the components and (c) specific services provided by his cloud provider. To achieve that goal, Player 1 has to:

- register his own components locally to the platform's private zone (through the registration tool of the platform) and connect them via semantic adapters;
- find other necessary components from the open shared space provided by third parties;
- provide a semantic description of the abstract components that will be on-the-fly transformed (discovered and invoked) to appropriate ones for the current context;
- create appropriate adapters to make cloud-specific services available through them;
- Provide a semantic annotation of the created Ecosystem Infrastructure.

Thus, using features of the platform such as: adapter-based connection of components (data sources and services), component discovery (based on their semantic specification), browsing of available resources based on their semantic description and the tool for semantic annotation of resources, Player 1 may create a PEI, annotate it and publish it to the marketplace.

Player 2 is a user of the PEI (provided by Player 1 or any other EI provider) and a service provider at the same time. The player would like to find an appropriate ecosystem infrastructure to create and launch his own application on top of it, as a publicly available service. An application is meant to be a dynamic self- configurable composition of several services. The way the application should work is context dependant. Among the relevant context variables are service availability, reliability, cost, and user and service location.

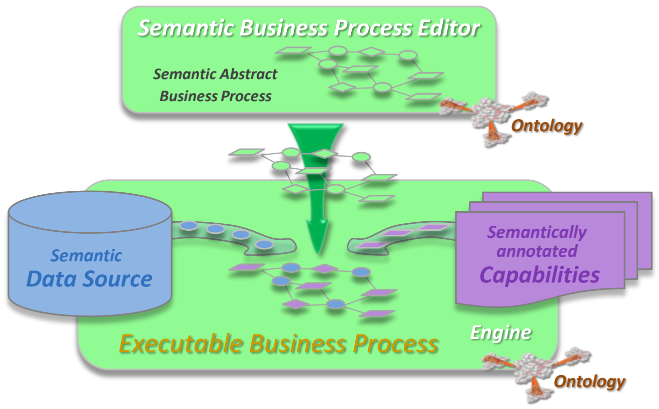
Semantic Business Process Editor
Semantic Abstract Business Process
Ontology
Semantic Data Source
Semantically annotated Capabilities
Executable Business Process
Engine
Ontology

Figure 3. Semantic Abstract Business Process of UBIWARE.

Player 2 enters the platform and selects or finds (via the corresponding tools of the platform) an appropriate Ecosystem Infrastructure which fits the requirements of the player depending on task and domain specifics of the planned service. Utilizing the infrastructure components of the corresponding ecosystem and the abstract process definition tool of the platform, Player 2 defines a partially abstract process. Components are described through their semantic annotations and will at run-time be selected among appropriate available components, depending on the context. Thus, concrete instances of the composed service will be built on-the-fly and executed by the platform engine. After Player 2 has published his/her application, it can be used by service users on the web.

IV. CONCLUSIONS

Within this paper we aimed at showing possible steps on how openness and interoperability of cloud ecosystems can potentially be achieved. To achieve this goal, we utilized the UBIWARE platform as a tool for proactive interoperability of distributed heterogeneous components. We presented an open environment for collaborative ecosystems with personalized cloud architecture in a sense of task- and domain-specific application development. We extend the cloud stack which allows us to organize collaboration between services and applications located in different clouds. All the components that can be considered as task and domain-specific are put to the Ecosystem Infrastructure layer (EIaaS). To increase flexibility and reliability of the applications and services created on top of PEI, we considered a semantic definition of Abstract Infrastructure. This way, it becomes possible to define context-dependent Data and services to be used by applications and services. Utilizing semantic web and multi-agent system approaches, the UBIWARE platform makes the ecosystem proactive and able to configure itself on-the-fly. This configuration happens depending on context and user needs, using the advantage of semantically adapted data, intelligent capabilities, and semantic abstract business process definition techniques.

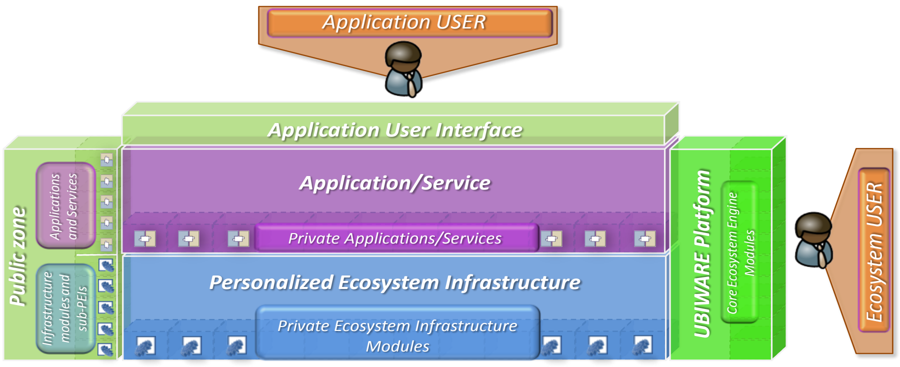
Application USER
Application User Interface
Application/Service
Private Applications/Services
Personalized Ecosystem Infrastructure
Private Ecosystem Infrastructure Modules
Public zone
Applications and Services
Infrastructure modules and sub-PEIs
UBIWARE Platform
Core Ecosystem Engine Modules
Ecosystem USER

Figure 4. General structure of the Ecosystem.

ACKNOWLEDGMENT

This research is based on activities of the project under the Cloud Software program in TIVIT SHOK (funded by TEKES and consortium of industrial partners) and the cCloud project (Academy of Finland) in the Department of Mathematical Information Technology (University of Jyvaskyla, Finland). We are very grateful to the members of the Industrial Ontologies Group for a fruitful cooperation within this research topic.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing”. Technical Report No. UCB/EECS-2009-28. EECS Department, University of California, Berkeley. February 10, 2009.

[2] D. Herris, “For Open Cloud Computing, Look Inside Your Data Center”, 2010 [Online] URL: http://gigaom.com/2010/03/28/for-open-cloud-computing-look-inside-your-data-center/ [Accessed 30 June 2011]

[3] Bernard Golden. (2009, January) Computer World. [Online]. URL: http://www.computerworld.com/s/article/9126620/The_case_against_cloud_computing_part_one [Accessed 30 June 2011]

[4] Distributed Management Task Force, (2009, June) “Open Virtualization Format White Paper”, Version 1.0.0, DSP2017.

[5] Semantic Web, 2001. [Online] URL: http://www.w3.org/2001/sw/ [Accessed 30 June 2011]

[6] T. Berners-Lee, J. Hendler and O. Lassila, “The Semantic Web”, Scientific American 284(5), 2001, pp. 34-43.

[7] Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D. L., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T. R. and Sycara, K. (2002) DAML-S: Web Service Description for the Semantic Web. In: International Semantic Web Conference (ISWC), June 9th - 12th, Sardinia, Italy. pp. 348-363.

[8] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. 2002. “Importing the Semantic Web in UDDI.” In Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web (CAiSE '02/ WES '02), Christoph Bussler, Richard Hull, Sheila A. McIlraith, Maria E. Orlowska, Barbara Pernici, and Jian Yang (Eds.). Springer-Verlag, London, UK, UK, 225-236.

[9] FIPA, “FIPA Interaction Protocol Library Specification Specification”, FIPA00025, 2001. URL: http://www.fipa.org/specs/fipa00025/ [Accessed 30 June 2011]

[10] F. Curbera, M. Dufler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, “Unraveling the Web Services Web: An introduction to SOAP, WSDL and UDDI”, Internet computing, 2002.

[11] F.M.T.Brazier, J.O. Kephart, H. Parunak and M.N. Huhns, "Agents and Service-Oriented Computing for Autonomic Computing: A Research Agenda," IEEE Internet Computing, vol. 13, no. 3, pp. 82-87, May/June 2009, doi:10.1109/MIC.2009.51

[12] N. Jennings, “An agent-based approach for building complex software systems”. Communications of the ACM 44, 4, 2001, pp. 35–41.

[13] N. Jennings, “On agent-based software engineering”, Artificial Intelligence 117(2), 2000, pp. 277–296

[14] UBIWARE Project [Online] URL: http://www.cs.jyu.fi/ai/OntoGroup/UBIWARE_details.htm [Accessed 30 June 2011]

[15] O. Kaykova, O. Khriyenko, D. Kovtun, A. Naumenko, V. Terziyan and A. Zharko, ”General Adaption Framework: Enabling Interoperability for Industrial Web Resources”, In: International Journal on Semantic Web and Information Systems, Idea Group, ISSN: 1552-6283, Vol. 1, No. 3, July-September 2005, pp.31-63.

[16] F. Bellifemine, G. Caire, A. Poggi and G. Rimassa, “Jade, A White Paper” [Online]

URL: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf [Accessed 30 June 2011]

[17] A. Katasonov and V.Terziyan, “SmartResource Platform and Semantic Agent Programming Language (S-APL)”, In: P. Petta et al. (Eds.), Proceedings of the 5-th German Conference on Multi-Agent System Technologies (MATES’07), 24-26 September, 2007, Leipzig, Germany, Springer, LNAI 4687 pp. 25-36.

[18] O. Khriyenko, S. Nikitin and V. Terziyan, “Context-Policy-Configuration: Paradigm of Intelligent Autonomous System Creation”, In: Joaquim Filipe and Jose Cordeiro (Eds.), Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS-2010), 8-12 June, 2010, Funchal, Madeira - Portugal, ISBN: 978-989-8425-05-8, pp. 198-205.

[19] P. Mell, T. Grance, “The NIST Definition of Cloud Computing (Draft)” Special Publication 800-145, January 2011.

[20] Amazon EC2, [Online] URL: http://aws.amazon.com/ec2/ [Accessed 30 June 2011]

[21] Microsoft Windows Azure [Online]

URL: http://www.microsoft.com/windowsazure/ [Accessed 30 June 2011]

[22] Amazon EC2 Publicly available datasets [Online] http://aws.amazon.com/publicdatasets/ [Accessed 30 June 2011]