On MDA-SOA based Intercloud Interoperability framework

Tahereh Nodehi[1], Sudeep Ghimire[2], Ricardo Jardim-Goncalves[3*], Antonio Grilo[4]

[1]UNL, t.nodehi@campus.fct.unl.pt,
[2]UNL, UNINOVA, sudeepg545@gmail.com,
[3]CTS, UNINOVA, rg@uninova.pt,
[4]UNIDEMI, acbg@fct.unl.pt
Faculdade de Ciências e Tecnologia da, Universidade Nova de Lisboa, Campus de Caparica, Monte de Caparica, Portugal

Abstract

Cloud computing has been one of the latest technologies which assures reliable delivery of on-demand computing services over the Internet. Cloud service providers have established geographically distributed datacenters and computing resources, which are available online as service. The clouds operated by different service providers working together in collaboration can open up lots more spaces for innovative scenarios with huge amount of resources provisioning on demand. However, current cloud systems do not support intercloud interoperability. This paper is thus motivated to address Intercloud Interoperability by analyzing different methodologies that have been applied to resolve various scenarios of interoperability. Model Driven Architecture (MDA) and Service Oriented Architecture (SOA) method have been used to address interoperability in various scenarios, which also opens up spaces to address intercloud interoperability by making use of these well accepted methodologies. The focus of this document is to show Intercloud Interoperability can be supported through a Model Driven approach and Service Oriented systems. Moreover, the current state of the art in Intercloud, concept and benefits of MDA and SOA are discussed in the paper. At the same time this paper also proposes a generic architecture for MDA-SOA based framework, which can be useful for developing applications which will require intercloud interoperability. The paper justifies the usability of the framework by a use-case scenario for dynamic workload migration among heterogeneous clouds.

Keywords: Cloud Computing, Intercloud, Cloud Interoperability, Model Driven Architecture (MDA), Service Oriented Architecture (SOA).

1. Introduction

Cloud computing as a recent computation paradigm has been developing very quickly. A cloud delivers on-demand services ranging from software to platform or infrastructure services (SaaS, PaaS, and IaaS) over the internet. To date cloud environments include hundreds of individual, heterogeneous, private/hybrid clouds with finite physical resources, but it is predicted in the near future expansion of the application scope of cloud services requires cooperation between the clouds. This interworking mechanism between clouds is called "Intercloud". Interoperability between clouds can provide:

- Better Quality of Service such as scalability and better reliability, service availability and performance than a single cloud system.
- Avoidance of vendor lock-in by using multiple clouds and freely migrating workload among them.
- Enabling inter-cloud Resource Sharing and enabling cloud users to use combined services from different service providers. These widely distributed resources can also reside in data centers worldwide.
- Reducing power consumption and/or labor costs due to delivering services from various locations.
- Currently, there are no implicit interoperability standards for heterogeneous cloud computing architectures to promote Intercloud interoperability. Different cloud computing systems from various companies and even the government are usually not interoperable.

Based on literature review, Service Oriented Architecture (SOA) method can considerably improve the cloud computing environment to provide the required service models with agility and scalability [1]. Additionally, according to literature, combination of Model Driven approach from Object Management Group (OMG) and SOA methodology can be exploited to perform analysis, design and implementation of enterprise integration and enhanced interoperability. According to the research focused on this paper, developing a novel

[*] Corresponding author: rg@uninova.pt

framework based on Model Driven Architecture (MDA) and SOA approaches, can improve Intercloud Interoperability. Additionally, this paper also discusses the concept and state-of-the-art in Intercloud Interoperability. In order to have a better understanding of MDA and SOA technologies, an introduction to MDA Models (CIM, PIM, and PSM), model transformations concept and languages, metamodel concept are included. Furthermore, this document includes basic concept of SOA method and description of core standards of SOA approach to provide interoperability. Finally a scenario is described in order to define the case study for our Intercloud Interoperability Framework which exploits FI-WARE platform that has a cloud hosting generic implementations. The FI-WARE[1] cloud offers Generic Enablers (GEs)[2] with the aim of establishing a modern cloud hosting infrastructure to develop and manage Future Internet applications and services. One of these GEs is JobScheduler GE. Our framework is according to JobScheduler GE to dispatch different tasks dynamically over multiple computing resources from other cloud providers.

2. Cloud Computing

The concept of "Cloud" is not a new one and it has been used in several fields such as ATM networks in 1990s. The term of "Cloud" is used to describe the networks that incorporate various technologies, without the user knowing it.

In 1997, as the first academic definition, Chellapa clarified cloud computing as "a computing paradigm where the boundaries of computing will be determined rationale rather than technical" [2].

The National Institute of Standards and Technology (NIST) proposed a cloud computing definition as follows: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models" [3][4].

Fig. 1. shows the framework introduced by NIST to define cloud computing [5].

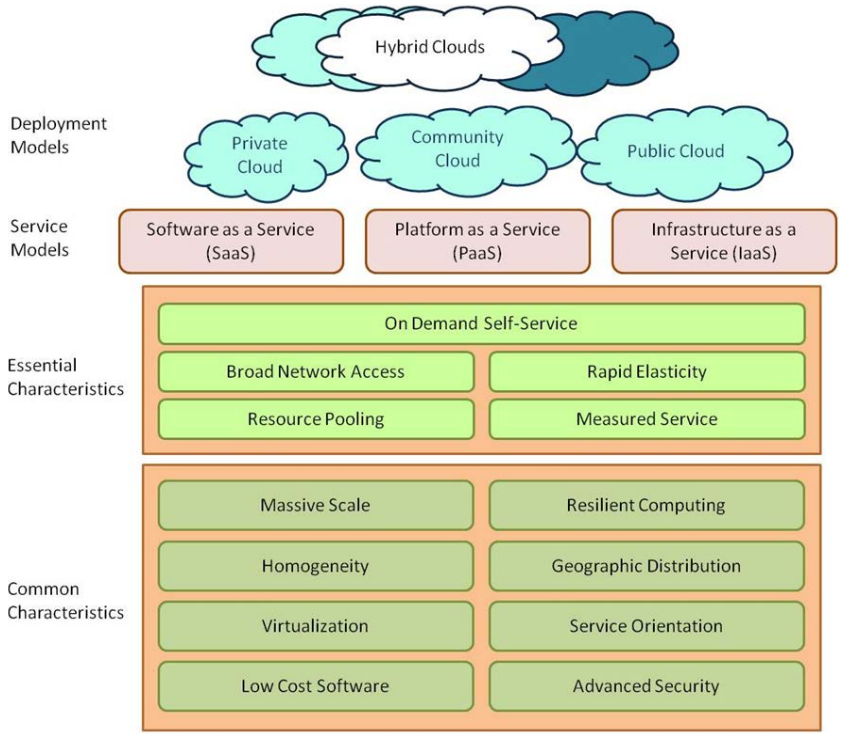


Fig. 1. Cloud Computing Definition [5].

[1] http://www.fi-ware.eu/
[2] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Cloud_Hosting

According to the different perspectives of various corporations such as; academicians, architects, consumers, developers, engineers and managers, there are several definitions for cloud computing [6]. This document adopted the cloud definition from Bernstein et al [7] article:

"Cloud Computing is a datacenter which:

- Implements a pool of computing resources and services which are shared amongst subscribers.
- Charges for resources and services using an "as used" metered and/or capacity based model.
- Are usually geographically distributed, in a manner which is transparent to the subscriber (unless they explicitly ask for visibility of that).
- Are automated in that the provisioning and configuration (and de-configuration and unprovisioning) of resources and services occur on the "self service", usually programmatic request of the subscriber, occur in an automated way with no human operator assistance, and are delivered in one or two orders of seconds.
- Resources and services are delivered virtually, that is, although they may appear to be physical (servers, disks, network segments, etc) they are actually virtual implementations of those on an underlying physical infrastructure which the subscriber never sees.
- The physical infrastructure changes rarely. The virtually delivered resources and services are changing constantly."
- According to the NIST [4] definition, cloud computing specifies three delivery models to provide various services such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS):
- The IaaS vendors provide a scalable, secure, and accessible infrastructure over the Internet [8]. Amazon's EC2 [9], Windows Azure Virtual Machines [10], and Rackspace Cloud [11] are some popular available IaaS.
- Fundamentally, PaaS provides a high level of abstraction to allow developers to focus on building higher level applications. Software developers can provide a custom developed application without bothering customers with managing and maintaining the infrastructure. Google Compute Engine [12], AWS Elastic Beanstalk [13] and Microsoft Azure are popular PaaS examples.
- *SaaS* is a cloud computing layer where users access applications running on a cloud infrastructure and offered on a platform on-demand [6][4]. Usually the users are able to run these applications using a client interface, like a web-browser. Practically, all of the underlying implementation and deployment is abstracted from the SaaS clients and only a specific set of configuration controls are accessible. Furthermore, the relevant data of SaaS applications is transparently placed in the cloud infrastructure. Google Apps [14], Salesforce [15], SuccessFactors [15] are popular SaaS examples.

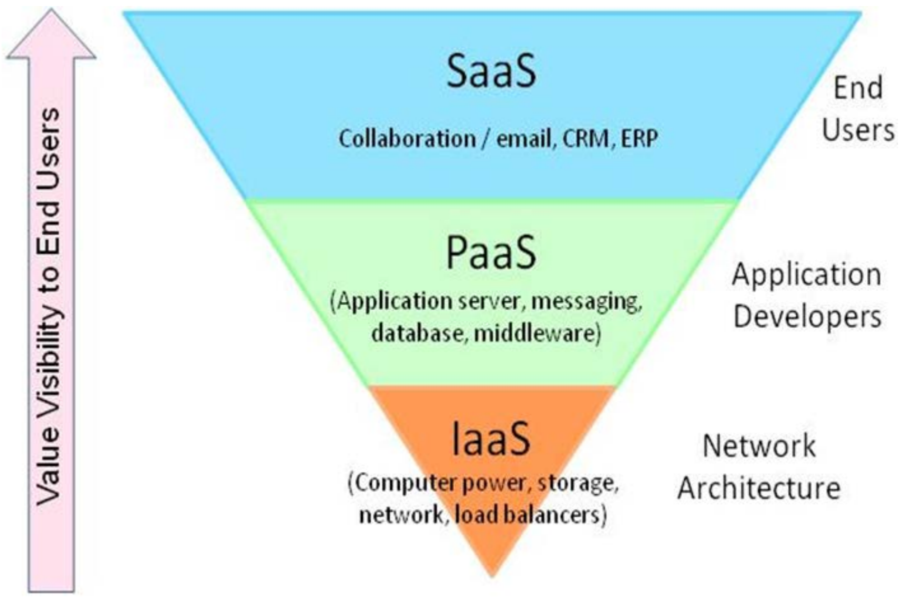

Value Visibility to End Users
SaaS
Collaboration / email, CRM, ERP
End Users
PaaS
(Application server, messaging, database, middleware)
Application Developers
IaaS
(Computer power, storage, network, load balancers)
Network Architecture

Fig. 2. Cloud Computing Pyramid.

Each level of service models adds additional functionality and provides required services for different kind of users from network architectures to end users (shown in Fig. 2.). Vaquero and colleagues introduced a flowchart (Fig. 3.) to illustrate the different actors and service delivery layers in Cloud Computing.

Zhang and colleagues [16] proposed a four layered architecture covering the three level of service model in cloud computing. As shown in Fig 4., the architecture includes the hardware/datacenter layer, the infrastructure layer, the platform layer and the application layer.

- The *hardware layer* is in charge of the physical resources available in the cloud, such as physical servers, routers, power and cooling systems. The hardware layer is normally implemented in the datacenters.
- The *infrastructure layer,* known as the *virtualization layer* is a crucial part of cloud computing. Its main responsibility is providing a pool of storage and computing resources by logical partitioning of the physical resources using virtualization technologies like Xen [17], KVM [18] and VMware [19].
- The *platform layer* is made up of operating systems and application frameworks to optimize running applications in VM containers.
- The *application layer* includes the cloud applications that can trigger the auto-scaling feature to achieve better performance, availability and lower operating cost.

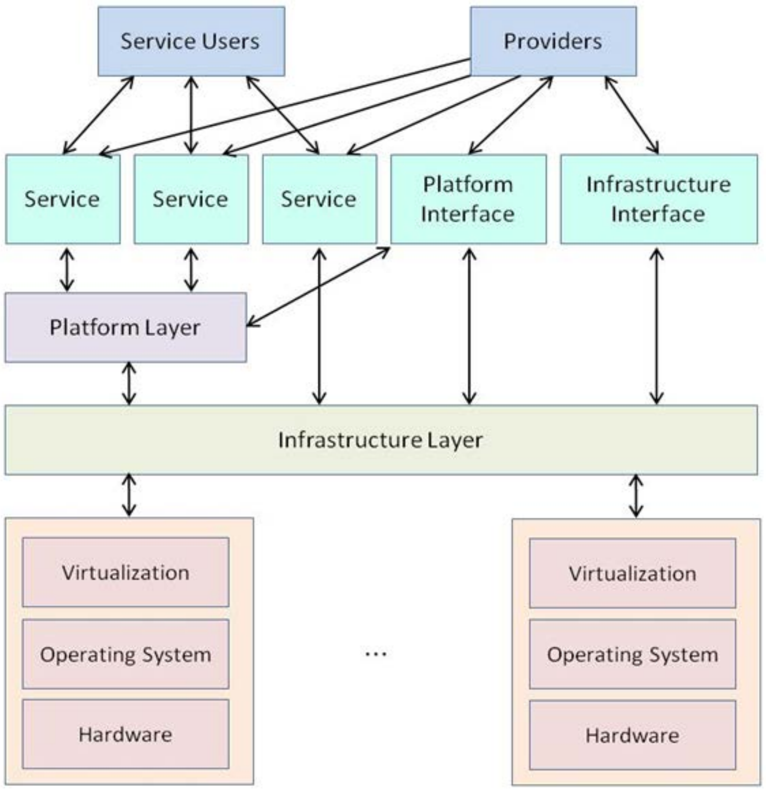


Fig. 3. Overview of actors and layers in Cloud Computing [80].

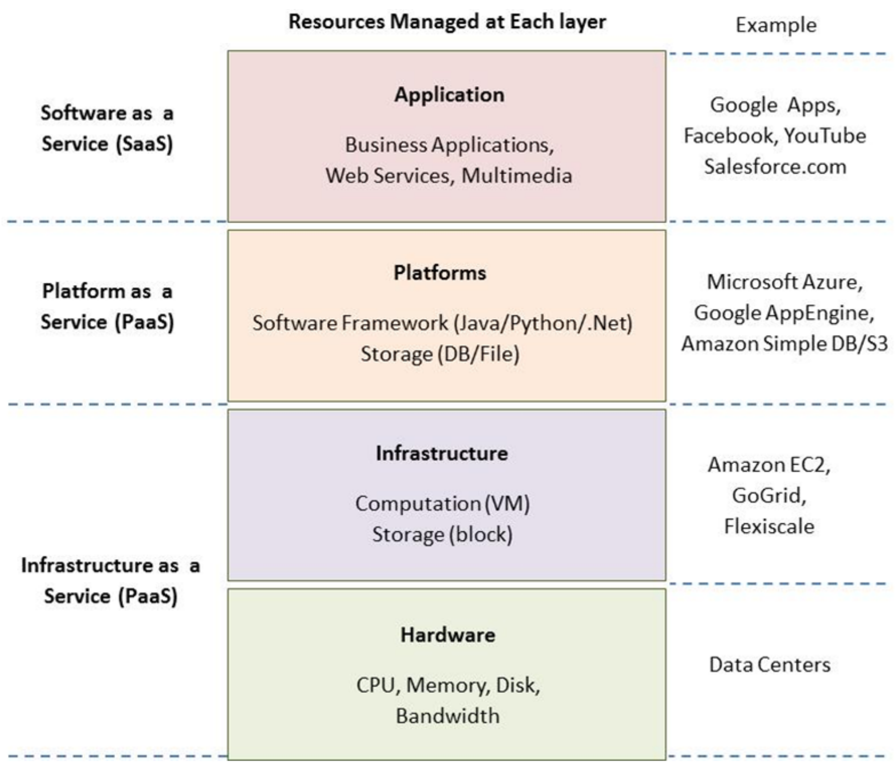

Resources Managed at Each layer
Example
Software as a Service (SaaS)
Application
Business Applications, Web Services, Multimedia
Google Apps, Facebook, YouTube Salesforce.com
Platform as a Service (PaaS)
Platforms
Software Framework (Java/Python/.Net) Storage (DB/File)
Microsoft Azure, Google AppEngine, Amazon Simple DB/S3
Infrastructure as a Service (PaaS)
Infrastructure
Computation (VM) Storage (block)
Amazon EC2, GoGrid, Flexiscale
Hardware
CPU, Memory, Disk, Bandwidth
Data Centers

Fig. 4. Cloud computing architecture [16].

There are four generic types for cloud computing infrastructure deployment (shown in Fig. 5.): public cloud, private cloud, community cloud and hybrid cloud [20][6][4][5][21]. The architecture, the datacenter's location, and the requirements of cloud customers determine different deployment strategies [22].

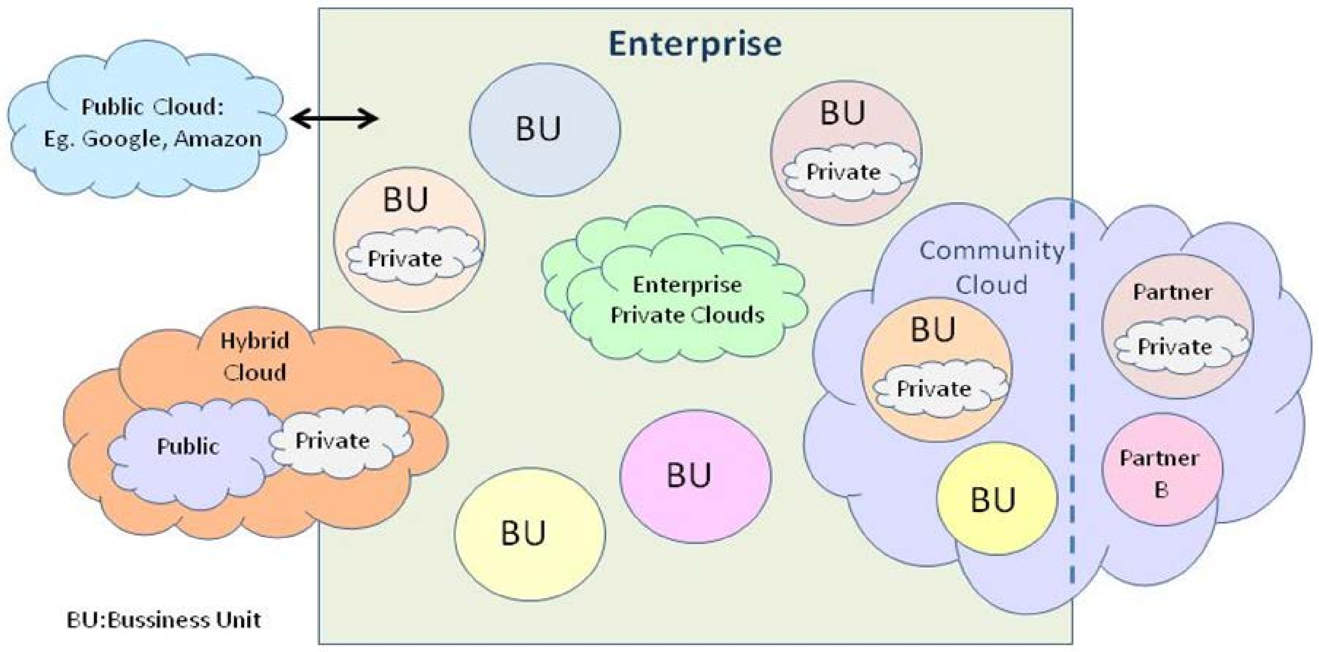


Fig. 5. Deployment models.

2.1. Intercloud Interoperability

Currently, cloud computing is an emerging computation paradigm in information technology and networking. Although Cloud Computing shared services has been increasingly utilized by diverse users, the research on Cloud Computing is still at an early stage. There are many existing cloud challenges that have not been fully addressed in addition to the new emerging issues introduced by enterprise applications. One of the existence challenges is the Intercloud Interoperability issue.

Intercloud became popular in early 2009 [23][7][24]. The Intercloud concept is based on the fact that each single cloud has limited computing resources in a restricted geographic area. Intercloud addresses the interoperability between various cloud computing instantiations where each cloud would use computing resources of other clouds. Cloud Computing environments need to be interoperable in order to reduce scaling/producing cost within the development of the components. Cloud costumers should be able to migrate in and out of the cloud and switch between providers based on their needs, without a lock-in which restricts customers from selecting an alternative provider. Furthermore, cloud providers should be able to interoperate among themselves to find an alternative cloud provider to give better services. The present Intercloud network merely connects different cloud systems and each cloud provider has its own way on how cloud applications/customers interact with the cloud. Feldhaus [25] summarized the current challenges in Cloud Interoperability as follow:

- Several different Cloud Standards from different parties are available.
- Existing Open Grid Forum (OGF) standards not or only partly ready for the cloud.
- A consistent OGF Cloud Portfolio is needed.
- Strategies for combining different Cloud Standards / APIs are needed.
- Existing implementations of Cloud APIs need to get interoperable.
- Combined Interoperability Verification Suites need to be developed.
- It is essential to discuss on issues related to specifications and implementation.

Currently different organizations, such as IEEE, are working on developing essential standards and appropriate APIs for Intercloud Interoperability. The future Intercloud network will expand the required functions to prepare collaboration among cloud services. Grozev & Buyya summarized their studies and classified 20 major Intercloud developments including both academic and industry projects [26]. According to their studies, Intercloud is classified as (Error! Reference source not found.Fig. 6.):

- *Volunteer federation*: when there is voluntarily collaboration between cloud providers that is often feasible for governmental clouds or private cloud portfolios.
- *Independent:* when an application or its broker independently from the cloud providers (both governmentally and private clouds) exploit multiple clouds.

Volunteer federation is classified in two architectural categories (Fig. 7) [26]:

- *Centralised*: there is a central entity in this architecture for intercloud to perform or facilitate resource allocation.
- *Peer-to-Peer*: *in this architecture, each cloud cooperates with the others directly*.

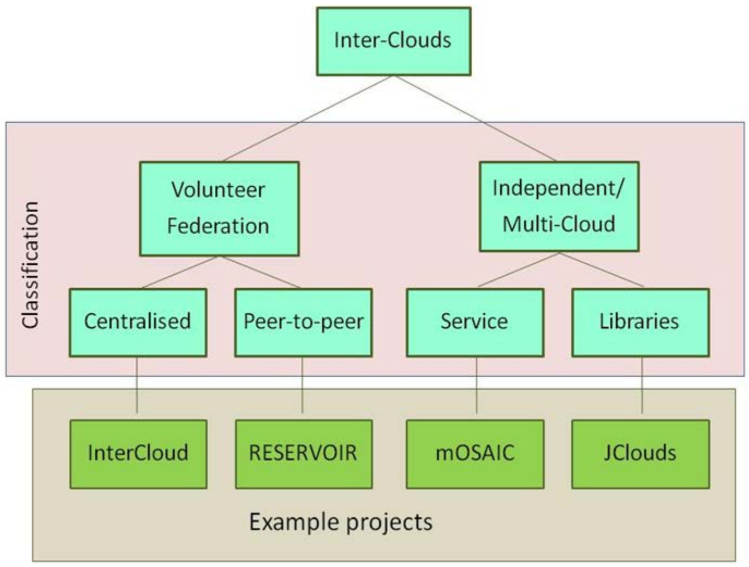


Fig. 6. Architectural classification of intercloud [26].

Furthermore, the *Independent Intercloud* development is classified in two architectural categories (Fig. 7.) [26]:

- *Services:* a service hosted externally or in-house by the users provides the application. Often, broker components are part of this type of services, and an SLA or a set of provisioning rules for application developers are defined by application and the service executes in the background according to the predefined attributes.
- *Libraries:* usually custom application brokers are required to provide and schedule application components directly across clouds. These approaches exploit intercloud libraries which facilitate utilizing multiple clouds uniformly.

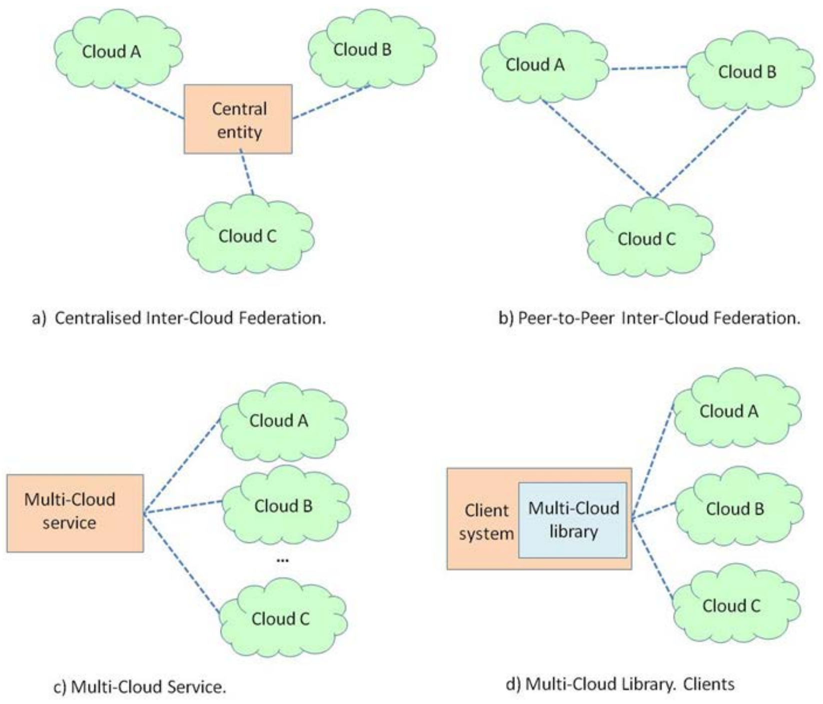


Fig. 7. Intercloud developments' architecture.

Dillon and colleagues [27] summarized some key intentions to solve the interoperability issue in the Cloud environments:

- *Intermediary Layer*: Providing an intermediary layer between cloud users and cloud computing resources (e.g.VM) may help improving cloud systems' interoperability. For instance, an abstraction layer can be developed at a higher level to provide a single resource usage model, user authentication model and API to support heterogeneous cloud providers.
- *Standard:* Standardization can be a solution to address the interoperability problem. The consensus between existing cloud providers, such as Amazon, Microsoft, or Google, is a big problem that makes standardization process very intricate.
- *Open API:* Open cloud API can define a set of clear and simple web services interfaces, to allow consumers to create and administrate cloud resources, including compute, storage, and networking components in a unified way.
- *SaaS and PaaS Interoperability*: Cloud providers mostly focused on IaaS interoperability problems, and few studies have highlighted interoperability issues in the other service deployment models.

Bernstein and colleagues [7] defined "Intercloud vision" shown in Fig. 8. to depict that various services from heterogeneous cloud systems are interoperable. Reference topology in Fig. 9. shows of how clouds interact in an InterMany of standards from current Internet networks are appropriate standards to reuse in Intercloud.

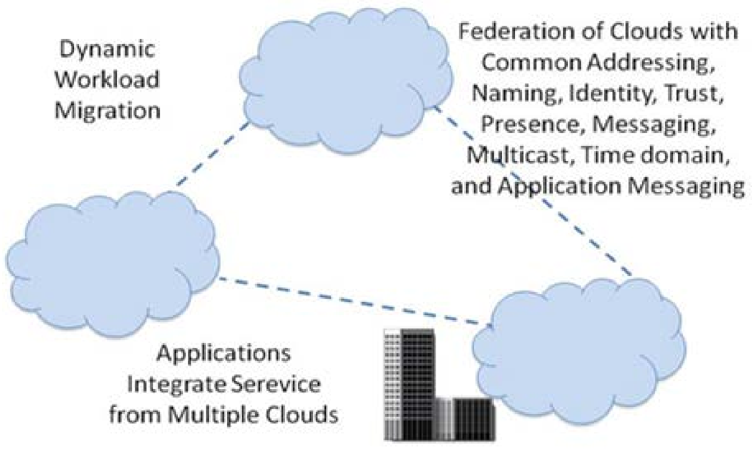

Dynamic Workload Migration
Federation of Clouds with Common Addressing, Naming, Identity, Trust, Presence, Messaging, Multicast, Time domain, and Application Messaging
Applications Integrate Serevice from Multiple Clouds

Fig. 8. The Intercloud Vision [7].

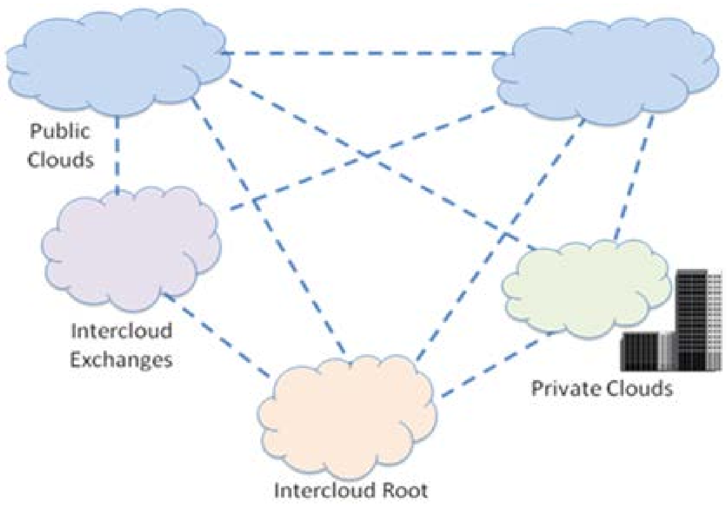


Fig. 9. Reference Intercloud Topology [79].

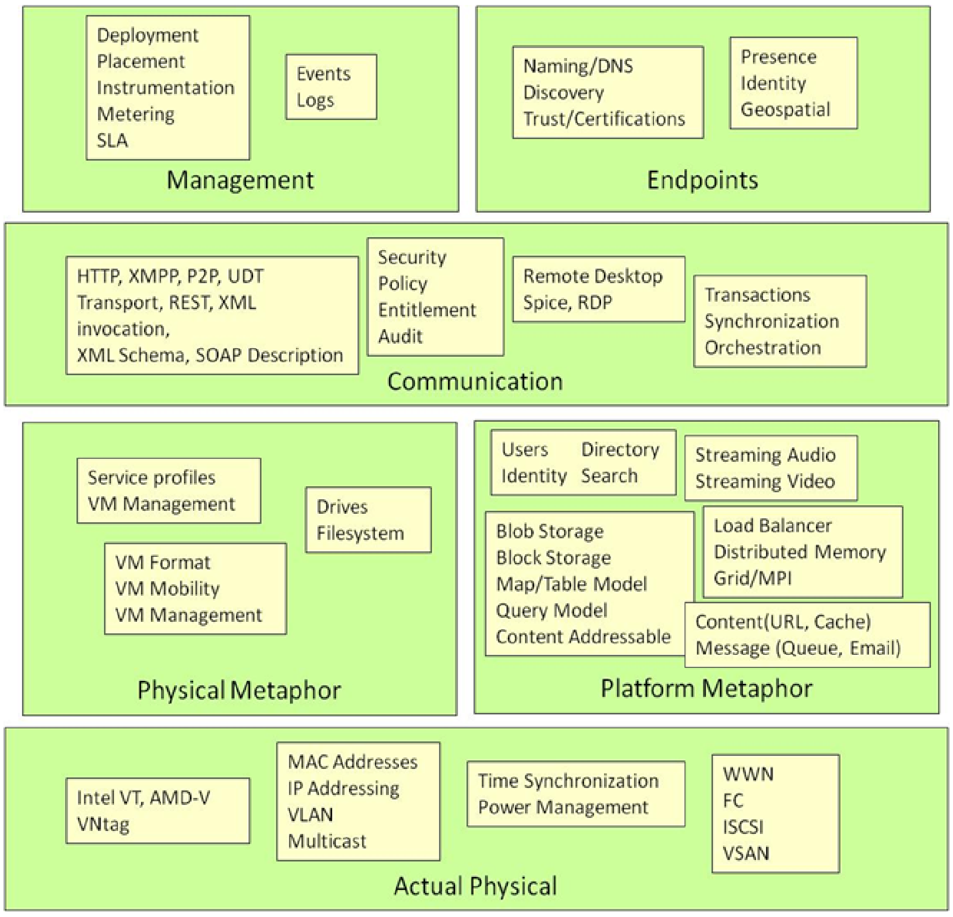


Fig. 10. An Architecture for Intercloud Standards [7].

Parameswaran and Chaddha [24] explained and examined two approaches in order to provide Intercloud standards and interoperability view:

Approach 1: Unified Cloud Interface/Cloud Broker and approach.

Approach 2: Enterprise Cloud Orchestration Platform /Orchestration layer.

Recently, the IEEE P2302 group [28] has been focusing on cloud-to-cloud interface standards for Intercloud Interoperability and Federation. Celesti in 2010 [29] proposed a three-phase (discovery, match-making, and authentication) cross-cloud federation model. This model represents an architectural solution (with some restrictions) to provide interoperability. In July 2009 in Japan, the Global Inter-Cloud Technology Forum (GICTF) published Intercloud Protocol [30] and Resource Data Model [31] to recognize the operational requirements of Intercloud systems and describe a peer-to-peer intercloud interface. However, it has been claimed in [32] Point to Point protocols are not appropriate for Intercloud Protocols and accordingly many-to-many mechanisms including Messaging and Presence Protocol (XMPP) for transport, and Semantic Web techniques such as Resource Description Framework (RDF) as a way to specify resources have been proposed. Bernstein and colleagues [32] used an XMPP Java API for a Cloud Service. Celesti also selected XML based technologies like XMPP to address interoperability issues [29]. Bernstein and colleagues [7] collected protocols, standards, formats, and common mechanisms as a beneficial architecture to implement Intercloud interoperability (Fig. 10.).

The Intercloud network scenario is still in an early stage. It needs more research work to provide sufficient functions to enable collaboration between cloud services. We are planning to present a framework to develop Intercloud Interoperability using two key technologies, MDA and SOA, described in following sections.

3. The Model Driven Architecture (MDA) Approach

The Object Management Group (OMG) announced the Model Driven Architecture (MDA) initiative as a software development approach to system-specification and interoperability based on the use of formal models [33]. MDA focuses on the development of models rather than detailed, platform-specific code which can be generated when needed. Instead of requiring developers to define every detail of a system's implementation using a programming language, it lets them model what functionality is needed and what overall architecture the system should have.

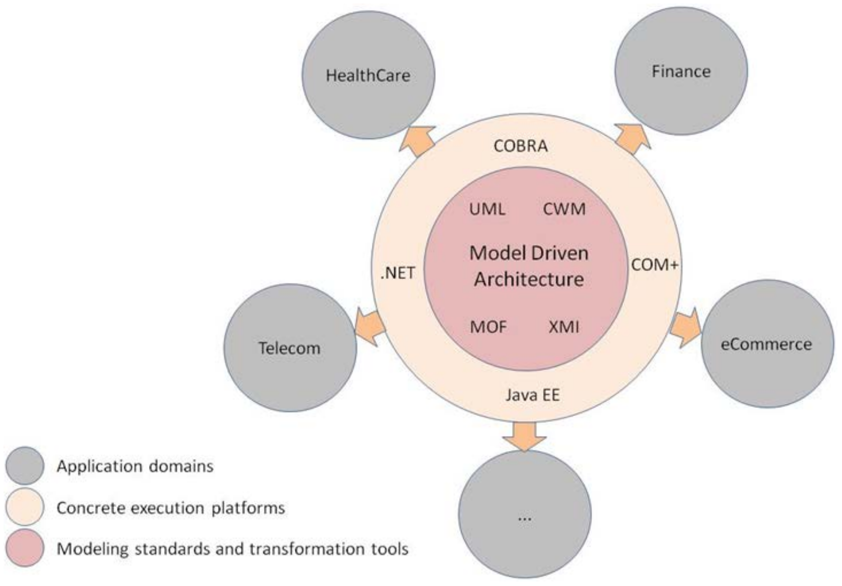

HealthCare
Finance
COBRA
UML
CWM
Model Driven Architecture
.NET
COM+
MOF
XMI
Telecom
eCommerce
Java EE
...
Application domains
Concrete execution platforms
Modeling standards and transformation tools

Fig. 11. The Model Driven Architecture [34].

The MDA approach gives the facility to understand complex and real-world systems while providing an abstraction of the physical system as shown in Fig. 11. [34]. This abstract view of the system is represented through the OMG's modeling standards including the Unified Modeling Language (UML) [35], Meta-Object Facility (MOF) [36], Common Warehouse Metamodel (CWM) [37] ,and XML Metadata Interchange (XMI) [38] which facilitates automatic generation of an XML-based document for a model according to its MOF definition.

MDA specifies three level of modeling abstractions: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM) (Fig. 12.).

The *Computational Independent Model* (CIM) represents what the business actually does or wants to do in future, but hides all information technology related specifications to remain independent of how that system will be implemented. CIM is independent from the use of the system as a computer system, and excludes any implementation details [39]. In other words, this model could be viewed as a contractual element that acts as a reference to check if client requirements have been correctly fulfilled.

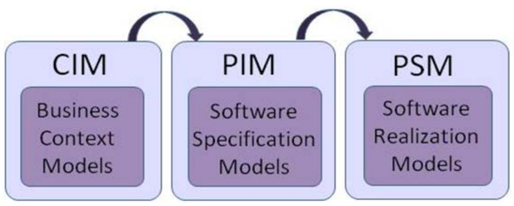


Fig. 12. Model Driven Architectures levels.

Ideally, software application design should be appropriate for all type of execution platforms (different operating systems, hardware, network protocols, programming languages, etc.) To achieve this Platform Independent Model (PIM) has been defined which provides a formal definition of the functionality of software system without addressing any specific operating platform. A platform-independent modeling language, such as UML, is used to design PIM model. The PIM model defines data, dependencies and architectural realizations. The model elements should provide enough information to make accordant code generation possible in next step. Based on platform independent model,

Platform specific model (PSM) provides the details to specify how the system uses a particular type of platform. In other words, PSM intends to ease generating corresponding code from the PIM that fits the operating platform [34]. The PIM model describes the system independent of XML, WSDL, SOAP, UDDI, Java, and other implementation technologies. The model-to-model and model-to-code transformations process would be accomplished using transformation tools that generate XML, WSDL, SOAP, UDDI, and the technology-specific artifacts and finally the implementation code from the design input [40].

Transformation techniques play a key role in making Model Driven approach successful. Transformations can be categorized based on the type of source and destination they operate on. At top level, model transformation approaches can be identified as *model-to-code* transformations or *model-to-model* transformations. Refer [41] for details on classification of MDA transformations. Various *transformation languages* and tool suites have been developed, such as QVT (Query/View/Transformation) [42][43][44], ATL (ATLAS Transformation Language) [45], GReAT (Graph Rewrite And Transformation language) [46][47][48], JTL (Janus Transformation Language) [49][50], Model-to-Model (M2M) [51], and MOLA: (MOdel transformation Language) [52][53][54].

4. The Service Oriented Architecture (SOA) Approach

Service Oriented Architecture (SOA) is a new architectural style to develop applications through services. It is defined as a collection of independent services which communicate with each other. The communication can include a simple data passing or two or more services coordinating the same activity. The connection for exchanging request and subsequent response messages between service customer and provider are specified in an understandable way to both the service consumer and provider. SOA is a new paradigm for solution architects to facilitate developing new value-added solutions by incorporating different solution artifacts such as business processes, services, packaged applications, and manageable attributes all over their lifecycle [55].

SOA defines an interaction model between three main functional units, shown in Fig. 13., in which the service consumer identifies adequate service via communication with the service provider through searching registry [56]. Practically, SOA contains six entities in its conceptual model, described as follow [56]:

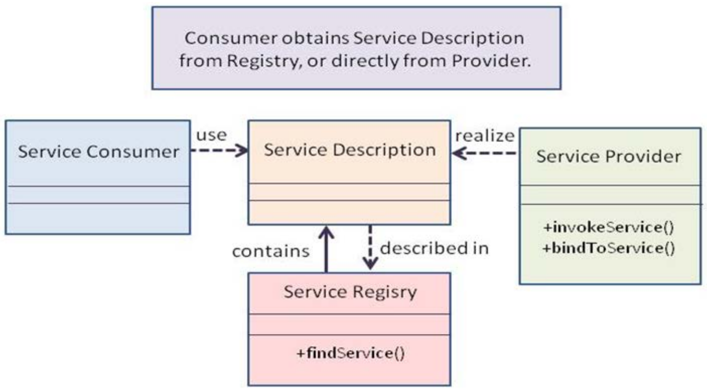


Fig. 13. Service Oriented Architecture Conceptual Model [56].

- *Service Consumer*: It is the entity that requests a service to execute a demanded function. If consumer knows the location of the service, it can communicate directly with the service provider, otherwise, it can detect the service location through the registry.
- *Service Provider*: It is an addressable entity of network that receives and executes the requests of consumers. It can provide the determined service description and the implement the service.

- *Service Registry*: It is a directory for available services which can be exploited through network. Service Registry should be able to publish and save service descriptions from providers and deliver the descriptions to the interested service consumers.
- *Service Contract*: It is a description that explicitly defines how the service consumer and provider should communicate. It includes information about the format of request-response message, the conditions in which the service should be executed, and quality aspects of the service.
- *Service Proxy*: It is an optional entity that facilitates the interaction between service provider and consumer through providing an API created in the local language of the consumer.
- *Service Lease*: It specifies and maintains the relationships between service consumer and provider. It defines the executive well-defined binding timeframes for the services that is managed by registry. It provides loose coupling between service provider and consumer as well as maintenance of state information for the service.

5. Interoperability via MDA and SOA

The interoperability between applications and services is inherent to the system design using MDA approach because MDA supports defining services, facilities, and applications through platform-independent model (PIM). Transforming the PIM to the PSM and then generating the code is based on the links provided between models. These links are specified by the metamodels' mappings (can also be linked with meta-models to add semantics) which allow platform specific and independent implementations to interoperate. Interoperability between two applications is provided by the mappings via the relevant metamodels of models. [57] and [58] have explored various dimensions of interoperability by making use of MDA and SOA. At the same time in the domain of the problem being addressed by this paper state of art on various transformation languages viz. ATL (ATLAS Transformation Language) [45], Model-to-Model (M2M) [51], and MOLA: (MOdel transformation Language) [54] provide a solid background to use MDA to achieve interoperability.

SOA inherits the ability of a service to be invoked by any potential service consumer and are connected using standard, dependency reducing decoupled message based methods. This methodology guarantees that services are coarse-grained reusable components that expose their functionality through a well-defined interface, systems can be built as a composition of services and evolve through the addition of new services. So, SOA methodology supports and promotes interoperable system designs. [59] presents a paradigm of cloud-marketplace ecosystem, making use of SOA to achieve collaborative marketplace architecture for the domain of e-procurement. At the same time oriented architecture Modeling Language (SoaML) offer several benefits such as [60] allowing service interoperability at the model level. A key issue for enabling interoperability is to come to an agreement about which services can be provided by whom and which can be consumed by whom in a network of service. Han at al. in [61] discusses how the OMG standards Business Motivation Model (BMM) [62] combined with SoaML can support Organizational Interoperability by enabling a community or organization to work together using SOA services at a higher level of abstraction. It also addresses service interaction concerns at the architectural levels by using architecture as the bridge between business requirements and automated IT solutions;

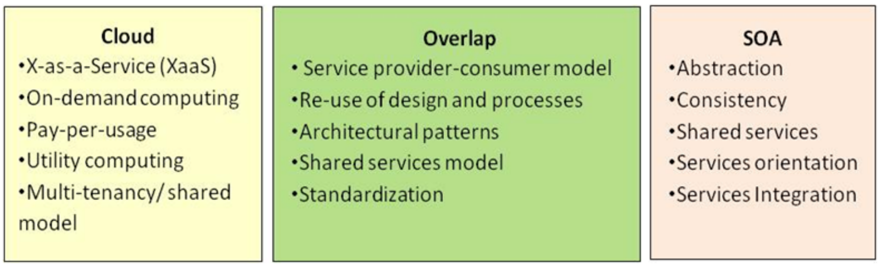
5.1 Current MDA, SOA based solutions for Cloud Computing

Recently, SOA and MDA approaches are increasingly exploited to develop different frameworks to alleviate several problems such as interoperability in enterprises [63][64][65][66][67][68][69]. Xu et al. claimed service interoperability is feasible using a model driven paradigm with service oriented systems [68]. Kim [69] specified main advantages to integrate a service-oriented modeling architecture with MDA:

- The clear organization of models and information based on the stereotypes derived from the SOA and Select Perspective as development process.
- The productivity, quality and impact analysis benefits of the use of MDA with its emphasis on automation, transformation and synchronization.

Cloud providers, mainly cloud Software-as-a-Service (SaaS), can use the advantages of MDA approach to develop the software applications. The interoperability between applications and services is the characteristic of a system designed based on MDA approach. Table 1 summarizes current research work on MDA-based solutions for Cloud Computing. Beside MDA approach, SOA method is a recent methodology which has significantly influenced IT architectures. SOA is fundamentally an architecture framework that can immensely help cloud computing architecture to provide the required services model with agility and scalability [1]. Additionally SOA promised interoperability between applications by put up application systems as group of published services [70]. Dillon and colleagues [27] described several ways that SOA can help implementing cloud services, such as Service Description for Cloud Services, Service Discovery for Cloud Services, Service Composition for Cloud Service, and Service Management for Cloud Service.

Considering the high-level definition of cloud and SOA, Infosys [1] presented how SOA and cloud overlap (Fig. 14.).



Cloud	Overlap	SOA
•X-as-a-Service (XaaS) •On-demand computing •Pay-per-usage •Utility computing •Multi-tenancy/ shared model	• Service provider-consumer model •Re-use of design and processes •Architectural patterns •Shared services model •Standardization	•Abstraction •Consistency •Shared services •Services orientation •Services Integration

Fig. 14. Cloud and SOA overlap in several architectural aspects [1].

Table 1 also shows the current research work on SOA-based solutions for Cloud Computing. In addition to leverage MDA or SOA based solutions separately to develop Cloud Computing, it is possible to merge SOA, and MDA in progress of optimal solutions for Cloud Computing (e.g Sharma's research work [71]). We are planning to exploit MDA-based SOA method to get the benefits of these technologies in implementing a novel framework for Intercloud Interoperability.

Table 1. State-of-the-art for MDA-based, and SOA-based solutions of Cloud Computing.

	Year	Title	Area	What had been done?
[72]	2011	Cloud SaaS and Model Driven Architecture.	MDA-Cloud	Incorporating MDA reduces the impact of applying software technological advancements on software applications and it augments the rigor, durability and reusability of the cloud services. In this paper, MDA approach was deployed to develop cloud SaaS.
[73]	2011	A Model-Driven Approach to Cloud SaaS Interoperability	MDA-Cloud	This paper introduced an MDA-based approach to provide interoperability among the software services in the cloud.
[74]	2011	Enhancing Cloud SaaS Development With Model Driven Architecture	MDA-Cloud	In order to have robust, flexible and agile software solutions for advanced cloud software applications, this paper studied the MDA approach to develop software systems
[1]	2011	Connecting the dots : Cloud and SOA	SOA-Cloud	Infosys released a whitepaper in 2011 to present the overlap between SOA and Cloud Computing and explain how SOA has being connected and enhanced cloud.
[75]	2012	SoaML and UPIA Model Integration for Secure Distributed SOA Clouds	SoaML-Cloud	This paper described the required information for SOA modelling techniques and some methods to exchange between U.S. Department of Defence (DoD) and commercial tools.
[71]	2011	Modelling Cloud SaaS with SOA and MDA	MDA-SOA-Cloud	This paper highlighted merging Cloud Computing, SOA, and MDA in progress of optimal business solutions.
[76]	2012	On-Demand Service-Oriented MDA Approach for SaaS and Enterprise Mashup Application Development	MDA-SOA-Cloud	This proposed an On-Demand Service-Oriented Model Driven Architecture approach that applies Service Oriented Architecture (SOA) elements into MDA to develop an enterprise mashup prototype.

6. Vision for MDA-SOA based Intercloud Interoperability

The aim of current article is introduce an Intercloud Interoperability framework based on SOA and MDA approaches for particular GE in FI-WARE Cloud. In order to show the distinctive ways of interaction between cloud users and providers, NIST [77] defined following *use cases for Cloud Computing Interoperability*:

- Copy Data Objects Between Cloud-Providers
- Dynamic Operation Dispatch to IaaS Clouds
- Cloud Burst from Data Center to Cloud
- Migrate a Queuing-Based Application
- Migrate (fully-stopped) VMs from One Cloud Provider to Another

Lewis [78] after studying use cases proposed by NIST and OMG, presented four main cloud interoperability use cases that can benefit from current standards:

1. User Authentication: A user who has established an identity with a cloud provider can use the same identity with another cloud provider.
2. Workload Migration: A workload that executes in one cloud provider can be uploaded to another cloud provider.
3. Data Migration: Data resided in one cloud provider can be moved to another one.
4. Workload Management: Custom tools developed for cloud workload management can be used to manage multiple cloud resources from different vendors.

The overall vision for MDA-SOA based inter-cloud interoperability to achieve the scenarios as explained is as shown in Fig. 15. A cloud based application makes use of the framework to interoperate with other clouds. Application accesses the functionality of the framework through the interfaces defined by the framework.

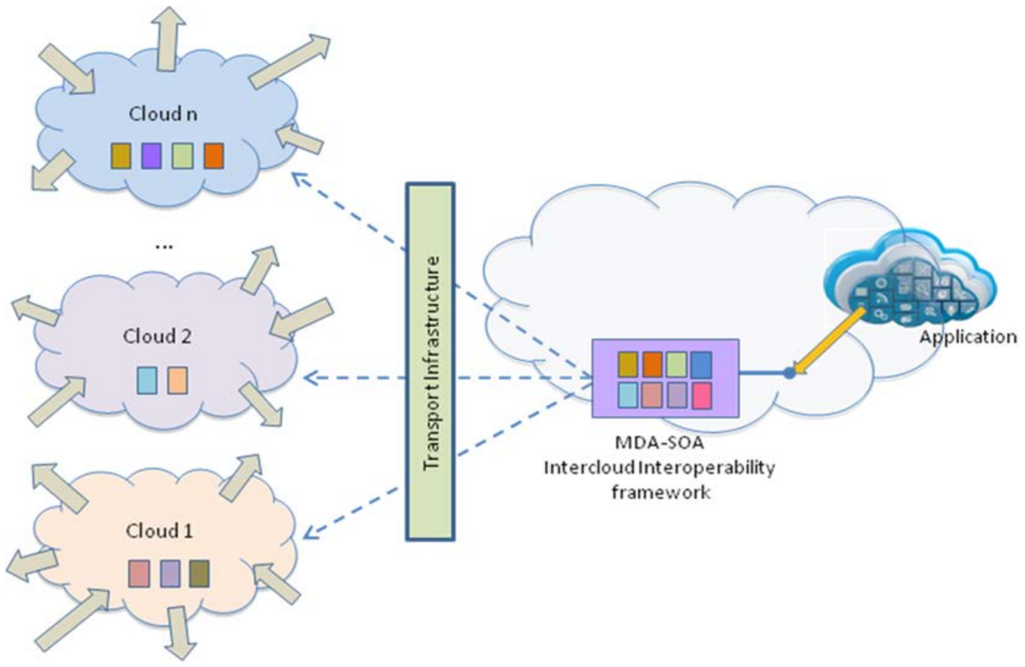


Fig. 15. MDA-SOA Intercloud Interoperability Framework.

XMPP is emerging as the transport protocol for inter cloud communication. In “Using XMPP as a transport in Intercloud Protocols” [32], Bernstein and colleagues has been claimed that Point to Point protocols are not advisable for Intercloud Protocols and accordingly many-to-many mechanisms including Messaging and Presence Protocol (XMPP) for transport are the appropriate way to communicate. MDA-SOA framework makes use of such standard communication protocol and other communication infrastructure as the Transport infrastructure. Transport Infrastructure is an important aspect of studies for inter-cloud interoperability, bus is out scope of this paper. In the following section, vision for MDA-SOA inter-cloud interoperability is described with further details.

6.1. Generic Architecture

The SOA-MDA inter-cloud generic architecture aims to resolve interoperability incompatibilities between heterogeneous Cloud computing Platforms. This architecture utilizes the knowledge driven from emerging IT trends such as MDA, SOA, semantics and also provides an interface for integration of other applications being developed to perform various tasks in the paradigm of cloud computing. The architectural pattern to be followed by the generic architecture is based on the discussion in sections 3 and 4. It comprises of two horizontal layers, the MDA-SOA Layer, the Enablers-Integration Layer and two vertical layers, namely the Semantics Layer and the Inter-Cloud Layer, that span across all the horizontal ones. A high-level view of the generic architecture is as shown in Fig. 16:

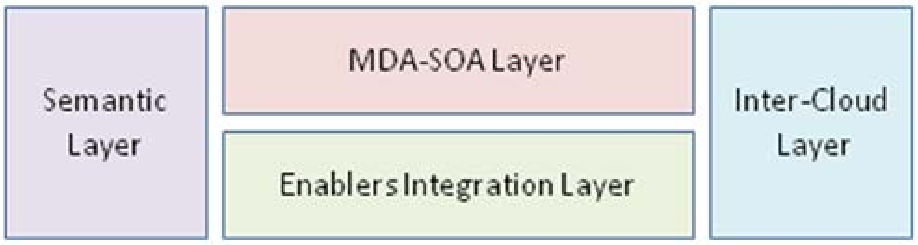


Fig. 16. A high-level view of the generic architecture.

The MDA-SOA Layer implements the core functionalities offered by the overall framework that will support major interoperability related operations. The *Enablers Integration layer* provides the interfaces for integration of third party cloud-based applications into the generic architecture, so as to achieve some specific tasks. The *Semantics Layer* provides the functionality to maintain and utilize the semantic models that will be necessary to obtain interoperability. The *Inter-Cloud Layer* puts in place the technical infrastructure related to independent clouds, which provides necessary information for all the horizontal layers. All of these functionalities will be exposed through well define interfaces like web service which provides an easy access for the MDA-SOA Framework functionalities.

6.1.1. Semantic Layer

The backbone layer of the architecture is the Semantic Layer (shown in Fig. 17.). Its components, named Application Model, Data Model and Cloud Offering Model, span the entire architecture resolving semantic interoperability conflicts that are raised between different clouds. Semantics are used by the MDA-SOA Layer in order to provide the means for developing interoperability related mechanisms.

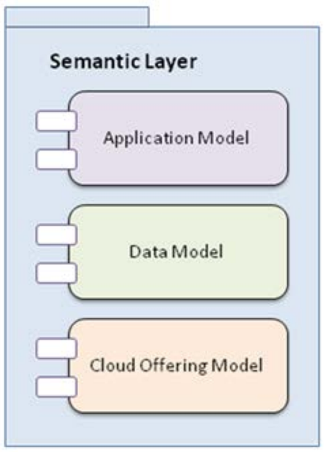


Fig. 17. Semantic Layer.

It is to be noted that Cloud Offering Model is the top level abstraction component to generalize different models of cloud offering. In any instance, this can be implemented by SaaS, PaaS or IaaS Offering Model, based on the use-case(s) for which the inter-cloud interoperability framework is being used.

6.1.2. MDA-SOA Layer

The MDA-SOA Layer, lies on top of the Enablers Integration Layer, and comprises of a components that will be accessible from top layer application interface layer, with well defined interfaces. Its components capitalize on the semantic annotation of the Semantic Layer and the functionalities of the inter-cloud layer to offer various cloud resources discovery and selection based on the requirements of the service consumer application which is obtained through the top layer i.e. application interface layer. At the same time, this layer makes use of enablers integration layer to achieve some specified tasks, based on the functionality provided by the enabler. On the whole, MDA-SOA layer acts as the mediator layer between all the other layers. This layer makes extensive use of the concepts and principles that have been discussed in sections 3 and 4. MDA-SOA Layer and its components are depicted in Fig. 18Error! Reference source not found.

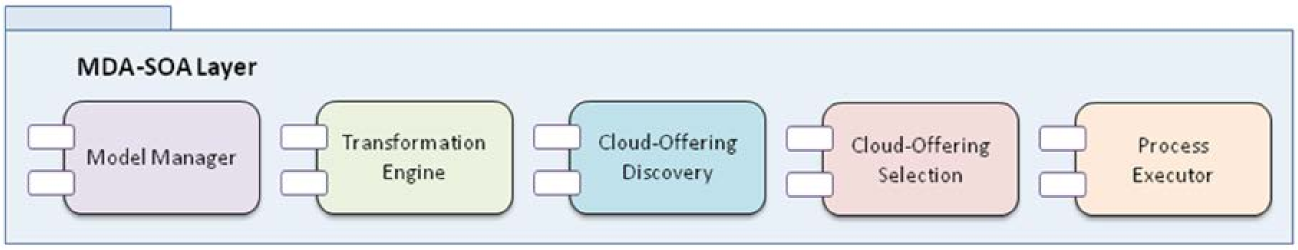


Fig. 18. MDA-SOA Layer.

- **Model Manager**: The Model Manager module uses the Cloud-Offering Model, the Application Model and the Object Model in order to retrieve the semantic concepts related to the corresponding object instances.
- **Transformation Engine**: Transformation Engine is responsible to perform model transformations based on the interoperability requirements. This engine makes extensive use of model manager to define the transformation strategy, based on the requirements of the clouds under interaction.
- **Cloud-Offering Discovery and Selection**: This component provides the functionality for cloud-offering discovery and selection from heterogeneous clouds. The Cloud-Offering Discovery and selection components capitalize on the search mechanisms and information offered by the inter-cloud Layer and employs lightweight semantic models and techniques in order to find among the available cloud offerings which meets the current work-flow requirements. These requirements are based on the models under considerations and the QoS specifications provided by service consumer that uses the framework. Note that this is a high level abstraction component which will be implemented by SaaS, PaaS or IaaS Offering Discovery and Selection components, as required by the application scenario.
- **Process Executor**: This component is primarily responsible for the execution of the business process, which defines the sequence of operations to be performed to achieve some specific task. In the architecture that we have proposed, we can observe that the components are separated and provide specialized functionality and also have the provision to integrate third party services/applications. So, this component is very important and it executes processes by interpreting them and evaluating their execution conditions. Every activity of the process model defined will be evaluated and the ones that satisfy the business conditions for the current work-flow would be executed.

6.1.3. Enablers Integration Layer

Interoperability between clouds will arise because of different use-case scenarios, which will require providing various implementations based on the problem domain. This, layer acts as the point of integration for such implementations which are termed as enables in this paper. So, the lower layer of the architecture provides an open space to integrate third party implementations, a example of which will be provided in section X as applicable for that particular scenario. The components being integrated in this layer virtually can be anything -service or application and will communicate with other layers or are used by other layers through well defined interfaces. So, in the generic architecture this layer is just an abstraction layer, and doesn't require any predefined components, because this layer doesn't implement any specific functionality.

6.1.4. Inter-Cloud Layer

One of the vertical layers of the generic architecture inter-cloud layer involves the appropriate capabilities that enhance the selection of specific providers form the network of cloud providers. Its main components support search and discovery mechanisms with the help of repositories. At the same time they support the selection mechanism by providing the profile of the cloud providers through QoS and SLAs repositories. This layer makes use of the SOA and Cloud computing principles as discussed in section 2. An abstract view of the Inter-Cloud Layer is presented in Fig. 19.

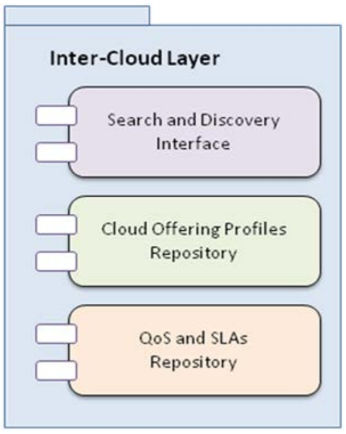

Inter-Cloud Layer
Search and Discovery Interface
Cloud Offering Profiles Repository
QoS and SLAs Repository

Fig. 19. Inter-Cloud Layer.

- **Search and Discovery Interface**: This component is responsible for searching among the available cloud offerings from the network of cloud providers. On the one hand there is a set of application requirements in the form of resources and restriction upon them and on the other hand there is a set of cloud offerings that express capabilities of the respective Cloud providers. The purpose of this component is to match the needs and offers producing a matching score.
- **Cloud Offering Profiles Repository**: This component serves as a simple registry for browsing over the available cloud offerings. A cloud offering is defined as a set of software, infrastructure or resources and is provided by one offering party. Such a party may provide more than one offerings.
- Note that both of these components are the high level abstraction components for cloud. Depending on the case of the use these will be inherited or implemented by IaaS, SaaS or Paas Search and Discovery Interface and IaaS, SaaS or Paas Offering Profiles Repository.
- **QoS and SLAs Repository**: A Service License agreement (SLA) is essentially a bridge between a cloud offering and application requirements set (Application Profile). It represents an agreement between the Cloud provider and the Cloud based application owner who is the consumer of the available services. Each SLA defines recovery actions if restrictions cannot be satisfied. At the same time QoS properties for each services of the cloud provider are provided by this repository which will be used for making the correct selection of the cloud provider (or the provided services) based on the requirements of the service consumer.

6.2. Scenario for job scheduling in Intercloud paradigm

In order to further explain the proposed framework, this paper selects “Workload Migration” as an interoperability use case, which is for workloads independent from unique resources of a specific cloud-provider and its task is dynamically dispatch the operations to the clouds. In, this scenario, Job Scheduler GE[3] from FI-WARE cloud hosting architecture can be integrated in the *Enablers Integration Layer*. The Job Scheduler GE is an enabler to execute a task over distributed multiple heterogeneous computer systems, both physical and virtual ones introduced in FI-WARE Cloud Architecture. Exploiting our Intercloud Interoperability framework for job scheduling GE will provide the job submission and its life-cycle control through available computing resources on some other cloud vendors.

Based on the scenario, an instance of the generic architecture is as shown in Fig. 20. In this scenario, the top level abstract components for cloud, are instantiated with the implementations for IaaS paradigm for cloud computing. The overall framework provides interfaces for job submission and model submission. Consumer application now interacts with the framework, the big picture of which is as shown in Fig. 15.

[3] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.JobScheduler

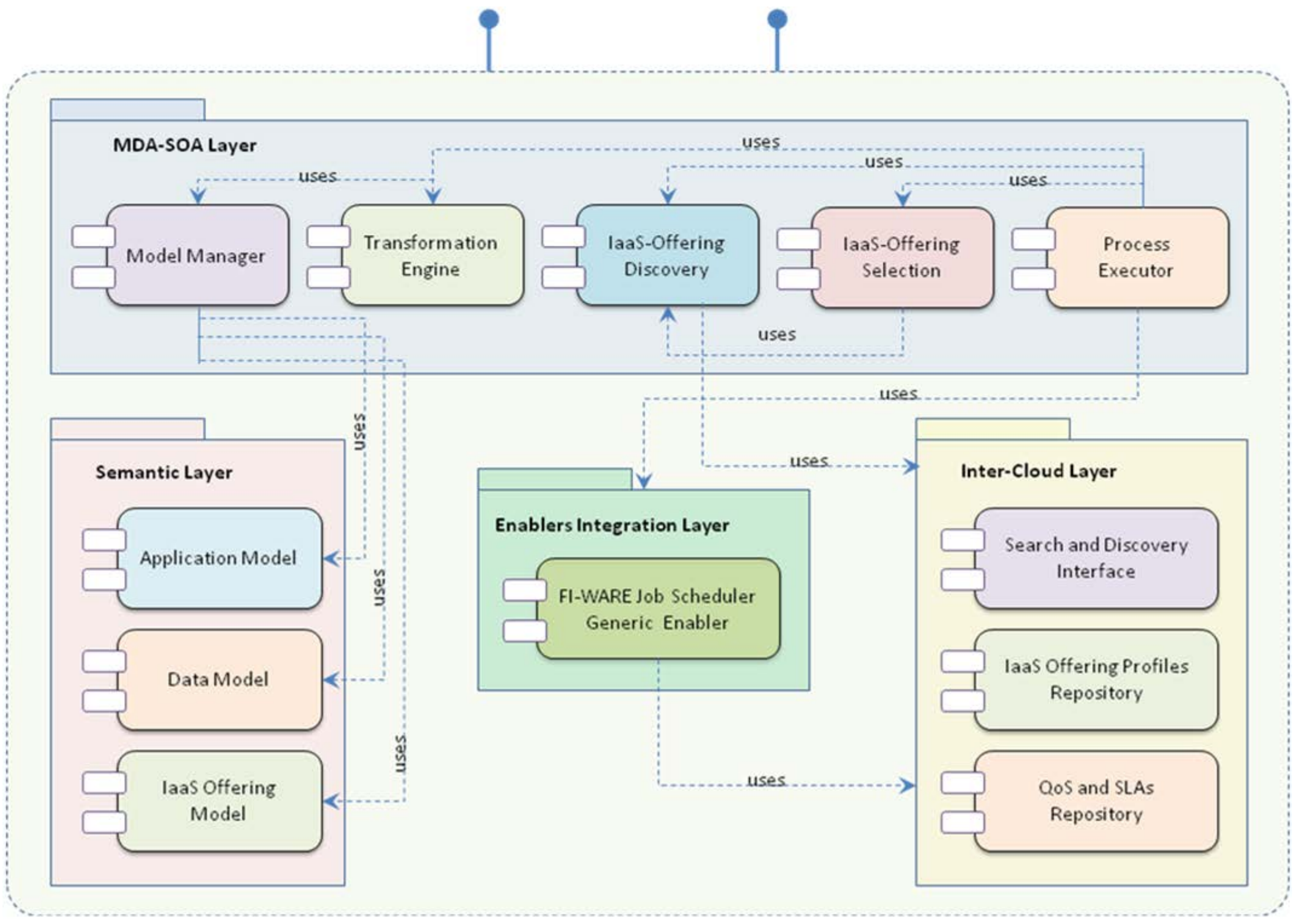


Fig. 20. Generic architecture for job scheduling in intra-cloud paradigm.

The sequence of activities that will take place can be depicted by the following steps:

1. Framework receives event for job dispatch from the service consumer, through its interface for consumer applications.
2. Service consumer provides the job details (Object Models, Operation Models etc.), through the service interface exposed by the framework, which is handled by the Model Manager module of the framework.
3. If applicable i.e. if the task is critical, service consumer also provides the QoS requirements for the task.
4. Upon reception of all the job details and necessary requirements, process executor engine initiates the work-flow and keeps track of all the activities.
5. Resource search and discovery module of the framework looks up for the available resources in other cloud and acquires the specified QoS and other functions specifications.
6. Resource selection module makes use of the requirements obtained in step 3 and provider specification obtained in step 5, to select the set of clouds that will be used for job dissemination. At this stage the IaaS offering selection module makes use of IaaS Offering Model to make the best suited selection.
7. Transformation Management performs the necessary model transformation for the job details obtained in step 2 as per the specification of the resources selected in step 6 and its details obtained in step 5. At this point the semantics layer will be used to make the necessary transformations.
8. Framework makes use of the Job scheduling GE to schedule the job to the selected resources in step 5 with the transformed model obtained in step 6
9. Selected resource executes the job and returns back the result.
10. Framework collects the results, performs necessary transformations (if necessary) and is sent back to service consumer.

7. Conclusion and future work

Cloud computing has emerged as a new and promising paradigm and includes managing heterogeneous, private/public/hybrid clouds and delivering services over the Internet. Many challenges exist in the area of Cloud Computing which can be an obstacle for its adoption by organizations to outsource applications with sensitive information over cloud environment. This paper tries to address one major challenge which is Intercloud Interoperability that is based on the fact that each single cloud has limited computing resources in a restricted geographic area. Intercloud Interoperability will enable cloud providers to deliver better quality of services, avoid data lock-in, and reduce scaling/producing costs.

Since, there is still no implicit solution to promote Intercloud Interoperability; we are working on a framework to achieve better Intercloud Interoperability. In order to devise the best approaches for implementation of our framework, current research approaches to Intercloud Interoperability, Cloud Computing, and different application design approaches were studied. Our research shows that recently, Model Driven approaches from OMG and SOA methodology are increasingly exploited to develop different frameworks to solve several issues such as interoperability in enterprises. This paper includes the main concepts in cloud computing and additionally the concept and state-of-the-art in intercloud interoperability. Moreover, in order to have better understanding of MDA and SOA approaches to implement Intercloud framework, this paper describes the capability of MDA and SOA approaches to enhance the interoperability among clouds as well as current state of the art in utilizing these approaches. This paper also proposes a generic framework based on the principles of MDA and SOA, to be used to resolve the intercloud interoperability issues. The generic architecture of the framework not only takes into account of the general MDA and SOA patters but also integrates other important aspects of interoperability like semantics and loose coupled third party services integration. The functioning of the framework is described by providing a specific instantiation of the generic architecture in the paradigm of the job migration into heterogeneous clouds. In this particular use-case, an implementation provided by FI-WARE is integrated into the framework to achieve some specific functionality.

This paper opens up lots of future work to be undertaken in the aspect of intercloud interoperability. One important future task will be to work on the development of the framework based in the concepts that have been proposed and study applicability in some real case business scenarios. Development of such framework will help in the adoption of intercloud by both cloud providers and consumers. At the same time other important future task that can be undertaken is to study other paradigms of inter-cloud as discussed in section 6 and resolve by using or extending the framework that has been proposed in this paper. A number of other cloud related generic enablers are being developed by FI-WARE project, which can be integrated with the proposed framework to study other paradigms where intercloud will be equally important and challenging.

Acknowledgement

The authors sincerely acknowledge the financial support from the grant of the Portuguese Foundation of Science and Technology (FCT) and EC 7th Framework Programme under grant agreement n° FITMAN 604674 (http://www.fitman-fi.eu).

References

[1] Infosys, *"Connecting the dots : Cloud and SOA"*, 2011.
[2] R. K. Chellappa, *"Intermediaries in Cloud-Computing: A new Computing Paradigm"*, Presented at the INFORMS meeting in Dallas, 1997.
[3] P. Mell and T. Grance, "*Perspectives on Cloud Computing and Standards*", National Institute of Standards and Technology (NIST), 2009.
[4] P. Mell and T. Grance, "*The NIST Definition of Cloud Computing Version 15*", National Institute of Standards and Technology (NIST), 2009.
[5] T. Grance, *"The NIST Cloud Definition Framework"*, National Institute of Standards and Technology (NIST), 2010.
[6] CSA, *"Security Guidance for Critical Areas of Focus in Cloud Computing V3.0"*, Cloud Security Alliance, pp. 0–176, 2011.
[7] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, *"Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability,"* 2009 Fourth International Conference on Internet and Web Applications and Services, 2009.
[8] GSA, *"Infrastructure as a Service (IaaS)"*, General Services Administration, 2012. [Online]. Available: http://www.gsa.gov/portal/content/112063.
[9] Amazon, *"Amazon Elastic Compute Cloud (Amazon EC2)*" [Online]. Available: http://aws.amazon.com/ec2/.
[10] Microsoft, *"Windows Azure, a rock-solid cloud platform for blue-sky thinking."* [Online]. Available: http://www.windowsazure.com/en-us/.
[11] Rackspace, *"Rackspace Cloud"* [Online]. Available: http://www.rackspace.com/cloud/.
[12] Google, *"Google Compute Engine"* [Online]. Available: https://cloud.google.com/products/compute-engine.

[13] Amazon, *"Amazon Web Services (AWS) Elastic Beanstalk"* [Online]. Available: http://aws.amazon.com/.

[14] Google, *"Google Apps : It provids independently customizable versions of several Google products under a custom domain name"*, 2011. [Online]. Available: http://www.google.com/apps/index1.html.

[15] and employees. Salesforce.com: An enterprise cloud computing leader that offers social and mobile cloud technologies that helps companies connect with customers, partners, *"Salesforce"* [Online]. Available: http://www.salesforce.com/eu.

[16] Q. Zhang, L. Cheng, and R. Boutaba, *"Cloud computing: state-of-the-art and research challenges"*, Journal of Internet Services and Applications, pp. 7–18, Apr. 2010.

[17] *"XenSource Inc, Xen"* [Online]. Available: http://www.citrix.com/products/xenserver/overview.html.

[18] *"Kernal Based Virtual Machine (KVM)"* [Online]. Available: http://www.linux-kvm.org/page/MainPage.

[19] *"VMWare ESX Server"* [Online]. Available: http://www.vmware.com/.

[20] M. Ahronovitz, D. Amrhein, P. Anderson, A. de Andrade, J. Armstrong, E. A. B, J. Bartlett, R. Bruklis, K. Cameron, M. Carlson, R. Cohen, T. M. Crawford, V. Deolaliker, P. Downing, A. Easton, R. Flores, G. Fourcade, T. Freund, T. Hanan, V. Herrington, B. Hosseinzadeh, S. Hughes, W. J. Huie, N. Q. Hung, P. Isom, S. R. J, S. Johnston, R. Kulkarni, A. Kunjunny, E. Lau, T. Lukasik, B. Marcus, G. Mazzaferro, C. McClanahan, M. Medley, W. Melo, A. Monroy-Hernandez, A. Nassar, D. Nicol, L. Noon, S. Padhy, G. Parann-Nissany, G. Pfister, T. Plunkett, L. Qian, B. Ramachandran, J. Reed, G. Retana, B. P. Rimal, D. Russell, M. F. Rutkowski, C. Sanford, K. Sankar, A. O. Sanz, M. B. Sigler, W. Sinclair, E. Sliman, P. Stingley, P. Straton, R. Syputa, R. J. Taylor, D. Tidwell, K. Walker, K. Williams, J. M. Willis, Y. Sasaki, M. Vesace, E. Windisch, P. Yara, and F. Zappert, *"Cloud Computing Use Cases White Paper, Version 4.0"*, 2010.

[21] D. Catteddu and G. Hogben, *"Cloud Computing: Benefits, risks and recomendations for information security"*, 2009.

[22] V. Delgado, *"Exploring the limits of cloud computing"*, KTH - Royal Institute of Technology, 2010.

[23] D. Bernstein, *"The Intercloud : Cloud Interoperability at Internet Scale"*, Sixth IFIP International Conference on Network and Parallel Computing, 2009.

[24] A. Parameswaran and A. Chaddha, *"Cloud interoperability and standardization"*, SETlabs briefings, 2009.

[25] F. Feldhaus, "Cloud Interoperability."

[26] N. Grozev and R. Buyya, *"InterCloud architectures and application brokering: taxonomy and survey"*, Software: Practice and Experience, 2012.

[27] T. Dillon, C. Wu, and E. Chang, *"Cloud Computing: Issues and Challenges"*,, 24th IEEE International Conference on Advanced Information Networking and Applications, Apr. 2010.

[28] *"IEEE P2302 Working Group (Intercloud)"*, [Online]. Available: http://grouper.ieee.org/groups/2302/.

[29] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, *"How to Enhance Cloud Architectures to Enable Cross-Federation"*, IEEE 3rd International Conference on Cloud Computing, Jul. 2010.

[30] [30] GICTF, *"Intercloud Interface Specification Draft (Intercloud Protocol)"*, Global Inter-Cloud Technology Forum, Whitepaper, 2012.

[31] [31] GICTF, *"Intercloud Interface Specification Draft (Cloud Resource Data Model)"*, Global Inter-Cloud Technology Forum, Whitepaper, 2012.

[32] [32] D. Bernstein and D. Vij, *"Using XMPP as a transport in Intercloud Protocols"*, 2nd USENIX Workshop on Hot Topics in Cloud Computing, 2010.

[33] OMG, *"MDA Guide Version 1.0.1"*, no. June, 2003.

[34] P. DE ALMEIDA, *"MDA – Model Driven Architecture : Improving Software Development Productivity in Large-Scale Enterprise Applications"*, University of Fribourg, Switzerland, 2008.

[35] OMG, *"Unified Modeling Language Specification Version 2.1.1"*, 2007.

[36] OMG, *"Meta Object Facility (MOF) Core Specification"*, 2006.

[37] OMG, *"Common Warehouse Metamodel (CWM) Specification"*, 2001.

[38] OMG, *"XML Metadata Interchange Specification Version 2.1"*, 2005.

[39] F. Chitforoush, M. Yazdandoost, and R. Ramsin, *"Methodology Support for the Model Driven Architecture"* 14th Asia-Pacific Software Engineering Conference (APSEC'07), pp. 454–461, Dec. 2007.

[40] D. S. Frankel, *"Model Driven Architecture Applying MDA to Enterprise Computing"* 2003.

[41] A. M. Jimenez, *"Change propagation in the MDA: A model merging approach"* The University of Queensland, 2005.

[42] OMG, *"Meta Object Facility (MOF) 2 . 0 Query / View / Transformation Specification"* 2011.

[43] OMG, *"MOF 2.0/XMI Mapping, Version 2.1.1"* 2007.

[44] D. van der Meij, *"A Metamodeling Approach to Incremental Model Changes"* University of Twente, 2009.

[45] *"ATL : Atlas Transformation Language ATL Starter's Guide"* no. December, ATLAS group LINA & INRIA Nantes, 2005.

[46] A. Agrawal, *"GReAT : A Metamodel Based Model Transformation Language"*

[47] A. Agrawal, G. Karsai, and F. Shi, *"Graph Transformations on Domain-Specific Models"* 2003.

[48] A. Agrawal, A. Vizhanyo, Z. Kalmar, F. Shi, A. Narayanan, and G. Karsai, *"Reusable Idioms and Patterns in Graph Transformation Languages"* Electronic Notes in Theoretical Computer Science, Mar. 2005.

[49] A. P. Antonio Cicchetti, Davide Di Ruscio, Romina Eramo, *"JTL - JANUS TRANSFORMATION LANGUAGE"* 2010. [Online]. Available: http://jtl.di.univaq.it/index.php.

[50] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio, *"JTL : a bidirectional and change propagating transformation language"* 3rd International Conference on Software Language Engineering, 2010.

[51] Eclipse, *"Eclipse Model-to-Model Transformation (M2M)"* 2006. .

[52] A. Kalnins, J. Barzdins, and E. Celms, *"Model Transformation Language MOLA : Extended Patterns"* Proceedings of Baltic DB&IS, Riga, Latvia, 2004.

[53] A. S. Audris Kalnins, Edgars Celms, *"Tool support for MOLA"* Fourth International Conference on Generative Programming and Component Engineering (GPCE'05). (Workshop on Graph and Model Transformation (GraMoT)) , Tallinn, Estonia, pp. 1–12, 2005.

[54] A. Kalnins, E. Celms, and A. Sostaks, *"Model Transformation Approach Based on MOLA 2 Brief Description of MOLA Language"*, ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML "2005). (MoDELS/UML"05 Workshop: Model Transformations in Practice (MTIP)) , Montego Bay, Jamaica, 2005.

[55] IBM, *"SOA"*, 2008. [Online]. Available: http://www-01.ibm.com/software/solutions/soa/.

[56] S. Güner, *"Architectural Approaches, Concepts and Methodologies of Service Oriented Architecture"*, Technical University Hamburg Harburg, 2005.

[57] R. Jardim-Goncalves, A. Grilo, and A. Steiger-Garcao, *"Challenging the interoperability between computers in industry with MDA and SOA"*, Computers in Industry, vol. 57, no. 8–9, pp. 679–689, Dec. 2006.

[58] R. Jardim-Goncalves, K. Popplewell, and A. Grilo, *"Sustainable interoperability: The future of Internet based industrial enterprises"*, Computers in Industry, vol. 63, no. 8, pp. 731–738, Oct. 2012.

[59] A. Grilo and R. Jardim-goncalves, *"Cloud-Marketplaces : Distributed e-procurement for the AEC sector"*, Advanced Engineering Informatics, vol. 27, no. 2, pp. 160–172, 2013.

[60] Y. Lemrabet, M. Bigand, and D. Clin, *"Model Driven Interoperability in practice : preliminary evidences and issues from an industrial project"*, First International Workshop on Model-Driven Interoperability (MDI 2010), 2010.

[61] F. Han, E. Moller, and A. J. Berre, *"Organizational Interoperability Supported through Goal Alignment with BMM and Service Collaboration with SoaML"* 2009 International Conference on Interoperability for Enterprise Software and Applications China, pp. 268–274, Apr. 2009.

[62] OMG, "Business Motivation Model (BMM) Specification, v1.0" 2008.

[63] A.-J. Berre, F. Liu, J. Xu, and B. Elvesaeter, "Model Driven Service Interoperability through Use of Semantic Annotations" International Conference on Interoperability for Enterprise Software and Applications China, Apr. 2009.

[64] C. Hahn, S. Jacobi, and D. Raber, "Enhancing the Interoperability between Multiagent Systems and Service-Oriented Architectures through a Model-Driven Approach" IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Aug. 2010.

[65] J. Touzi, F. Benaben, H. Pingaud, and J. P. Lorre, *"A model-driven approach for collaborative serviceoriented architecture design"*, International Journal of Production Economics, 2009.

[66] C. P. Bispo, R. S. P. Maciel, J. M. N. David, I. Ribeiro, and R. Conceicao, *"Applying a model-driven process for a collaborative service-oriented architecture"*, International Conference on Computer Supported Cooperative Work in Design, Apr. 2010.

[67] N. Ali, R. Nellipaiappan, R. Chandran, and M. A. Babar, *"Model driven support for the Service Oriented Architecture modeling language"*, International Workshop on Principles of Engineering Service-Oriented Systems - PESOS, 2010.

[68] J. Xu, Z. Bai, A. J. Berre, and O. C. Brovig, *"Model Driven Interoperability through Semantic Annotations using SoaML and ODM"*, Information Control Problems in Manufacturing, 2009.

[69] H. Kim, *"Modeling of Distributed Systems with SOA & MDA"*, IAENG International Journal of Computer Science, 2008.

[70] G. Shroff, Enterprise Cloud Computing Technology, Architecture, Applications. 2010.

[71] R. Sharma, M. Sood, and D. Sharma, *"Modeling Cloud SaaS with SOA and MDA"*, Advances in Computing and Communications - Communications in Computer and Information Science, 2011.

[72] R. Sharma and M. Sood, *"Cloud SaaS and Model Driven Architecture"*, ACCT: International Conference on Advanced Computing and Communication Technologies, 2011.

[73] R. Sharma and M. Sood, *"A Model-Driven Approach to Cloud SaaS Interoperability"*, International Journal of Computer Applications, 2011.

[74] R. Sharma and M. Sood, *"Enhancing Cloud SaaS Development With Model Driven Architecture"*, International Journal on Cloud Computing: Services and Architecture, 2011.

[75] R. W. Maule, *"SoaML and UPIA Model Integration for Secure Distributed SOA Clouds"*, IEEE Eighth World Congress on Services, Jun. 2012.

[76] X. Zhang, K. He, J. Wang, J. Liu, C. Wang, and H. Lu, *"On-Demand Service-Oriented MDA Approach for SaaS and Enterprise Mashup Application Development"*, International Conference on Cloud and Service Computing, Nov. 2012.

[77] L. Badger, R. Bohn, R. Chandramouli, T. Grance, T. Karygiannis, R. Patt-Corner, and J. Voas, *"Cloud Computing Use Cases," National Institute of Standards and Technology,*, 2010. [Online]. Available: http://www.nist.gov/itl/cloud/use-cases.cfm.

[78] G. A. Lewis, *"The Role of Standards in Cloud- Computing Interoperability"*, 2012.

[79] D. Bernstein, S. Clara, and N. Court, *"Using Semantic Web Ontology for Intercloud Directories and Exchanges"*, International Conference on Internet Computing, ICOMP'10, 2010.

[80] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, *"A Break in the Clouds : Towards a Cloud Definition"*, ACM SIGCOMM Computer Communication Review, 2009.