A Look at Cloud Architecture Interoperability through Standards

Claus Pahl, Li Zhang and Frank Fowley
School of Computing, Dublin City University, Dublin, Ireland
Email: [cpahl|lzhang|ffowley]@computing.dcu.ie

***Abstract*—Enabling cloud infrastructures to evolve into a transparent platform while preserving integrity raises interoperability issues. How components are connected needs to be addressed. Interoperability requires standard data models and communication encoding technologies compatible with the existing Internet infrastructure. To reduce vendor lock-in situations, cloud computing must implement universal strategies regarding standards, interoperability and portability. Open standards are of critical importance and need to be embedded into interoperability solutions. Interoperability is determined at the data level as well as the service level. Corresponding modelling standards and integration solutions shall be analysed.**

Keywords - Cloud Architecture; Interoperability; Standards.

I. Introduction

Enabling cloud infrastructures to evolve into a transparent platform while preserving integrity raises interoperability issues [2], [5]. Interoperability requires standard data models and communication encoding technologies compatible with the existing Internet infrastructure. The need to scale and provide cost-effective time-to-market. Public cloud services are available to the public and owned by an organisation selling cloud services, e.g. Microsoft or Amazon are major providers. Hybrid clouds are an integrated cloud services arrangement that includes provision of compute resources from more than one source (e.g. either private or public). Hybrid architectural models may be vertically partitioned (e.g. data stored privately) or horizontally partitioned (e.g. using public cloud to prototype a new device view of a service in parallel with an existing implementation). These architectural scenarios define the need for interoperability solutions if flexible composition, migration and portability are sought [1], [3].

To reduce vendor lock-in situations, cloud computing must implement universal strategies regarding standards, interoperability and portability. Open standards are of critical importance and need to be embedded into interoperability solutions [9], [10]. Standardisation efforts linked with intelligent processing techniques shall be given particular attention. Interoperability is determined at the data level as well as the service level [11], [4], [13]. Corresponding modelling standards and integration solutions shall be analysed.

The objectives of this investigation include the review of relevant standards for cloud architecture interoperability (looking at their background, usage and analysing their importance for this context) and analysing the overall maturity of the technology and determining current trends and shortfalls.

We start with an architectural scenario, the definition of stakeholders and interoperability concerns in Section 2. In Section 3, we categorise existing standards and review a selection, before ending with some discussions.

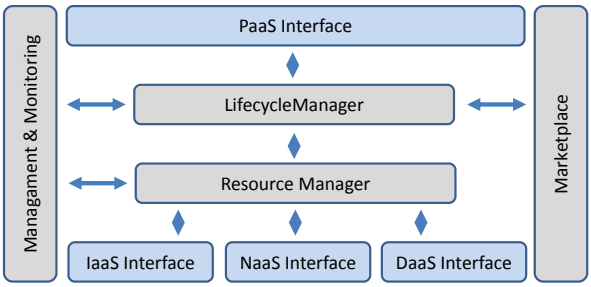

Fig. 1. Layered Architecture for Cloud-based Software Components.

II. Cloud Architecture and Interoperability

A. Cloud Architecture

A cloud architectural framework consists of the classical three cloud layers infrastructure (IaaS), platform (PaaS and software (SaaS) as service-oriented offerings [2], [5], [6]. In addition, we can differentiate between (hardware or software) resources provided in a traditional way, and the . . . *as a Service* version of them, which considers virtualization, multi-tenancy and elasticity as the concerns [3]. A platform product can be deployed over IaaS (or a network provided as a service NaaS) or over real hardware infrastructure. A platform product can be offered as a Service (PaaS) for the application layer. The application software can be deployed either on top of a platform product (cloud-less), or making use of platform services (PaaS). Finally, an application product can be offered as a Service (SaaS) for external customers.

Different usage models can be derived from the combinations of the layers. We take into account that some software (or hardware) is provides as a service, other components are directly interfaced: application over a platform, SaaS over a platform, or pure platform over IaaS. These different usage models rely on interoperability solutions. Some are service-based abstractions (APIs) that need, consequently, to be aligned with common (Internet/Web-based) service description, modelling and composition standards. However, also more technology (or layer) specific standards are also important.

A stakeholder can play more than one role within the platform scenario according to the usage. For instance, a Software Provider in the Application Software layer can be at the same time a PaaS or Platform Software customer. Service providers

and users rely more on service-related standards, whereas non-service interfaces for software providers and developers might be more layer-specific.

B. Stakeholder Roles

Different stakeholders can be associated with the architectural scenarios. These can be categorised into roles that reflect their activities and needs. These roles are based on suggestions from the EU FP7 Projects SLA@SOI (for general roles in the services context) and 4CaaSt (for cloud-specific roles).

A Service Provider (or application provider) supplies services to one or more internal or external customers. A Software Manager defines software-based services, takes care of their management (business focus) and administration (technical focus) A Service Aggregator is a reseller that contracts and aggregates services or applications by third parties in order to create a new ones A Service Maintainer maintains a service after it has been deployed A Context Provider provides context information about the underlying infrastructure components (e.g. telecoms or sensor network) A Data Provider owns, manages and provides data to a service

A Cloud Provider is a resource provider that provides an integrated platform or infrastructure services based on possibly heterogeneous cloud offerings. A Platform Provider offers a technical platform to Service Provider to host their software services. A Platform Manager manages a platform from a business perspective. An Infrastructure Provider is a cloud IaaS provider. An Infrastructure Manager measures and controls infrastructure properties.

A Software Provider produces software which might be used by a Service Provider to assemble services. In this context, a Software Designer designs/develops the architecture and components of a specific SLA-based application.

A Service Customer orders services and defines and agrees service-level targets (SLA). A Service Consumer is the person who actually consume/use the provided services.

C. Interoperability Concerns

Interoperability concerns arise in different situations. Interoperability between cloud layers needs standardised APIs to allow higher cloud layers to link to a range of services provided at the lower layers, e.g. platform implementations to uniformly link to IaaS offerings. Roles of importance are service provider and service user. Interoperability within layers needs suitable standards to allow components in a layer to interact and be exchangeable. Non-service interactions need to be supported, e.g. where, as explained in the third scenario above, a Software Developer combines different platforms in the development of a new system. Figure 1 indicates some of these components and their connectivity for the infrastructure and platform concerns. The architecture in Figure 1 should only be indicative of these concerns, but captures some agreed components in this context.

III. Interoperability-related Standards

Standards are necessary to consolidate efforts in a technology domain and to enable interoperability. An overview and a categorisation of standards relevant to interoperability in the cloud computing context that we cover here is:

- Web services: WSDL (description), SOAP (protocol), WS-BPEL (composition), UDDI (repository)
- Service modelling: Open-SCA (service composition and interaction), USDL/SoaML/CloudML (multi-view services), EMML (mashups)
- Service interfaces: OCCI (infrastructure management), CIMI (infrastructure management), EC2 (de-facto standard), TOSCA (portability), CDMI (data)
- Infrastructure: OVF (virtual machines); specific concerns: memcached (data caching), VEPA (network)
- Security: OAuth, SCAP

For each standard, we provide background about origins, support and purpose, the intended usage, and an analysis of the relevance for interoperability considerations. Providing a comprehensive overview of all standards is not the objective. We have singled out those that represent specific aspects well.

A. Core Web Services Standards

Service-based provision needs Web services alignment. Thus, relevant standards are SOAP, WSDL, WS-BPEL and UDDI (not discussed due to space considerations). As all cloud layers (infrastructure, platform, software, processes) can be provided in an . . .-as-a-service form, these classical services standards form the foundation of cloud interoperability. Although not standardised as such (and thus not covered here), RESTful services have become a similar, more lightweight major architectural style for services.

B. Service Modelling and Interface Standards

We separate more advanced modelling and interface description standards from the core Web services standards in this context, i.e. WSDL and WS-BPEL, as the ones covered here have not gained as much recognition.

Open Composite Services Architecture (Open-CSA) The OASIS Open Composite Services Architecture (CSA) specifications provide standards to simplify SOA application development [21]. OASIS brought together vendors and users from to collaborate on the development and adoption of the Service Component Architecture (SCA) and Service Data Objects (SDO) families of specifications. **Usage:** As an example, the SCA Assembly Model is a framework to describe service coordination and interaction that ties in service composition with common software architecture concerns. **Analysis and Recommendation:** The CSA standards can be utilised as is or can serve as input for any composition and assembly language for interoperability concerns. It can serve a guide for the specification of services and cloud configurations. It can also facilitate the development of visual tools for assembling of components and service references during application design.

The specifications on the SCA Assembly Model are very relevant for interoperability. Application development using SCA should result in the following advantages. It promotes decoupling of application business logic from the details of the invoked services. Target services in a range of languages

(like C++, Java, and PHP) are supported. Different communications constructs including One-Way, Asynchronous, Call-Return, and Notification are considered. Legacy components or services, accessed as Web Services, EJB, JMS, JCA, RMI, or CORBA can be included. Quality of Service requirements are considered, such as security, transactions and the use of reliable messaging

Data interoperability is an equally important concern in Open-CSA. Data could be represented in Service Data Objects.

The value proposition of SCA is, therefore, the flexibility for composite applications to incorporate reusable components in an SOA programming style. The overhead of business logic concerns regarding platforms, infrastructure, plumbing, policies and protocols are removed.

Enterprise Mashup Markup Language (EMML) Enterprise Mashups combine and remix data from databases, spreadsheets, websites, Web Services, RSS/Atom feeds, and unstructured sources that deliver actionable information. The Open Mashup Alliance (OMA) is in charge of the Enterprise Mashup Markup Language (EMML) [20]. It can support enterprise mashup implementations, improve mashup portability of mashup designs, and increase the interoperability of mashup solutions. OMA provides an EMML schema and a reference runtime environment as the technology framework. **Usage:** EMML is a Domain Specific Language (DSL) designed to address the creation and reuse of mashups. EMML is, however, not a general-purpose language - EMML was designed to be complimentary to and integrated with languages like JavaScript, Java, Groovy, and Ruby via scripting. EMML is a declarative XML-based language and, as such, leverages and complements existing XML capabilities inherent in XQuery, XPath, and XSLT. EMML is an open language specification. This free-to-use language (and technologies that embed or use it) have a much better chance of meeting the needs of enterprise developers than a proprietary language. **Analysis and Recommendation:** It is particularly suited for interoperability issues related to mashup creation. It is supportive of a strong trend towards lightweight and integrative content and service assembly and is therefore representative of a specific modelling and integration concern.

Unified Service Description Language (USDL) The aim of the Unified Service Description Language (USDL) team, an W3C Incubator Group, is to define a language for describing general, generic parts of technical and business services to allow services to become tradable and consumable [19].

- Technical services are considered software services based on WSDL, REST or other specifications.
- Business services are defined as business activities that are provided by a service provider to a service consumer to create value for the consumer.

The business services are more general and comprise manual and technical services. The USDL definition aims at complementing the technical language stack by adding required business and operational information. The targeted cloud stakeholders for USDL are service providers, infrastructure providers, service assemblers and service consumers. Industry-specific and general-purpose attributes of a service are derived based on use cases, taking into account the target groups. The USDL group aims to derive best practices and learning from testing cycles that can then be deployed in a number of use cases. These use cases serve as references and proof-of-concept of USDL. **Usage:** The language is usable for any purpose and implementation scenario of business services on a general level. However, it is also extendable for industry-specific aspects. USDL defines an interoperability-centric language that enables its users to model arbitrary services and to integrate with existing standards. **Analysis and Recommendation:** Particularly the aim to address service modelling and support this with mappings to different standards makes this a worthwhile framework for interoperable cloud service modelling. This enables new business models in the field of service brokerage because services can automatically be offered, delivered, executed, and composed from services of different providers. Business-IT alignment is an ongoing concern. Another development to be considered in this context includes SoaML (standardised by OMG, see http://www.omg.org/spec/SoaML/), which falls into the same category as USDL in our categorisation as a service description and modelling language. While still under development (and thus far from being standardised), CloudML (http://www.cloudml.org/) is a language more specific to clouds, developed by the same group as SoaML.

Open Cloud Computing Interface (OCCI)

OCCI (infrastructure lifecycle management) is now the first of four cloud-specific standards, also including CIMI (like OCCI on infrastructure management), TOSCA (portability and cloud-bursting), and CDMI (data management).

Cloud computing currently is organised into three models or layers offering Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), which all involve the on-demand delivery of computing resources. Providers offer IaaS solutions to enhance elastic capacity, where server instances are executed in their proprietary infrastructure and billed on a utility computing basis. For the infrastructure layer this means that typically virtual machines on a per-instance per-hour basis are the units. For the software (SaaS) layer, software application instances are the corresponding units, managed and billed with similar mechanisms. There are also both commercial and open source products that replicate this functionality in an in-house setting, but also exposing compatible interfaces as a hybrid cloud environments can be realised. The OGF OCCI working group provides an API specification for the management of cloud computing infrastructure [18]. **Usage:** The scope is a comprehensive range of high-level functionality for life-cycle management of virtual machines (or workloads) running on virtualization technologies (such as containers). OCCI provides an API for interfacing IaaS cloud computing facilities, which is sufficiently complete to facilitate the implementation of interoperable implementations:

- Consumers to interact with cloud computing infrastructure (e.g. deploy, start, stop, restart)
- Integrators to offer advanced management services
- Aggregators to offer a single common interface to multiple providers
- Providers to offer a standard interface that is compatible with available tools

- Vendors to offer standard interfaces for dynamically scalable service delivery in their products.

Analysis and Recommendation: OCCI is a step towards matching cloud-specific interoperability needs through standards. While targeting IaaS concerns, it can foster interoperability endeavours at higher levels. The scope of OCCI is high-level functionality for lifecycle management. This is in part realised through coverage of existing proprietary APIs. Storage details beyond creation and mapping of mount points is specifically excluded. Networking details are similarly excluded beyond creation and mapping of interfaces, assignment of these to public or private networks and assignment of dynamic or static IPs. While the focus is on the upper cloud stack layers for the section presented in Figure 1, it is nonetheless a suitable framework for interoperability at the interface of infrastructure services. OCCI allows, as an additional interoperability concern, the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring.

Cloud Infrastructure Management Interface (CIMI) Similar to OCCI, the CIMI - Cloud Infrastructure Management Interface from DMTF addresses infrastructure management. CIMI which addresses the runtime maintenance and provisioning of cloud services. The scope of the CIMI standard covers core IaaS functionality, addressing deploying and managing virtual machines and other artifacts such as volumes, networks, or monitoring. Once interfaced to the IaaS provider, the information that needs to be processed to manage a cloud service can be discovered iteratively, including the metadata describing capabilities and resource constraints. **Usage:** The model behind CIMI describes resources (systems or collections of resources managed as a whole, e.g. as an OVF file - which is covered below): machines (resource with CPU and memory), volumes (storage), and networks (representing layer 2 broadcasts). It also describes meters, which are metrics for some property, and event logs. Most developers use with the CIMI REST/HTTP-based protocol, the current interface binding to the model (others are expected later). This delivers standard HTTP status codes and supports JSON and XML serialization formats. **Analysis and Recommendation:** CIMI, if widely used, would allow organisations to design cloud-based business solutions being assured that management (and governance) processes will not be compromised if the business solution is moved to another (standards-based) IaaS provider.

Amazon Elastic Compute Cloud (EC2) While OCCI and CIMI are similar standards, in a wider context, Amazon EC2 as a proprietary solution and OpenStack as an open-source solution need to be considered in this context as well. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizeable computing capacity servers in Amazon's data centers. These can be used use to build and host software systems. EC2 follows a pay-as-you-go for the capacity that is needed. **Usage:** They allow access to components and features using a web-based GUI, command line tools, and APIs . At the core is an Amazon Machine Image (AMI), which is a template that contains a software configuration (operating system, application server, and applications). An AMI is used to instantiate (create) a virtual machine; it is an AMI is a filesystem image which includes an operating system (e.g., Linux, UNIX, or Windows) and any additional software required to deliver a service. From an AMI, instances are launched, which are running copies of the AMI. You can launch multiple instances of an AMI. Instances run until you stop or terminate them, or until they fail. **Analysis and Recommendation:** EC2 is a de-facto standard and comes with a rich ecosystem, including for instance monitoring tools such as CloudWatch or libraries for a range of programming languages. Some open-source standards are pushed by Amazon AWS competitors to regain market shares.

Topology and Orchestration Specification for Cloud Applications (TOSCA) Supported by OASIS, the TOSCA framework aims to enhance the portability of cloud applications and services. TOSCA enables interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behaviour of these services (such as deploy, patch, shutdown) independent of the supplier creating the service, and any particular cloud provider or hosting technology. TOSCA also aims to support higher-level operational behaviour to be associated with cloud infrastructure management. **Usage:** Through service and application portability in vendor-neutral settings, it enables:

- Portable deployment to any compliant cloud
- migration of existing applications to the cloud

thus adding to consumer choice and dynamic, multi-cloud provider applications. **Analysis and Recommendation:** The core concept behind TOSCA is cloud bursting, which is the ability to move workloads between public and private cloud infrastructures in a transparent way. There seems to be some discussion, with large IaaS providers not having joined the consortium yet. The core to the solution would be a hypervisor-agnostic portability mechanism, which requires IaaS compliance. TOSCA also needs to be observed as a vendor initiative in the context of open-source activities like OpenStack gaining momentum.

Cloud Data Management Interface (CDMI) The CDMI, the Cloud Data Management Interface by SNAI (see http://www.snia.org/cdmi), targets cloud storage. The Cloud Data Management Interface is a standard for self-provisioning, administering and accessing cloud storage. CDMI defines RESTful HTTP operations for accessing the capabilities of the Cloud storage system, including allocating and accessing containers and objects, managing users and groups, implementing access control, attaching metadata, making arbitrary queries, using persistent queues, specifying retention intervals and holds for compliance purposes, logging, billing, moving data between Cloud systems, and exporting data via other protocols. Transport security is via SSL/TLS. **Usage:** CDMI defines the functional interface that applications use to create, retrieve, update and delete data elements from the cloud. As part of this interface, a client can discover the capabilities of the cloud storage offering and use this interface to manage containers and the data that is placed in them. In addition, metadata can be set on containers and their contained data elements through this interface. **Analysis and Recommendation:** Compared to OCCI and OVF, CDMI specifically targets data moving and format immigration. Although CDMI can also be used for task management, this would need more extensive rules to be defined.

C. Infrastructure Standards

The OCCI is cloud-specific and addresses interoperability and interface concerns for the infrastructure level. We include here three further cloud standards that are specific to the cloud infrastructure level, of which OVF is the most critical and successful so far. This reflects the current activity and maturity in this context. Again, these are representative of a number of concerns and this selection is not meant to be exhaustive. Memcached and VEPA are representative of other concerns, but for instance as far as protocols are concerns, AMQP (Advanced Message Queuing Protocol) or STOMP (Simple (or Streaming) Text Orientated Messaging Protocol) could have been included.

Open Virtualization Format (OVF) The Open Virtualization Format (OVF), submitted to DMTF as a standard, describes an open, secure, portable, efficient, and flexible format for the packaging and distribution of one or more virtual machines [14], [16]. OVF features include:

- It enables optimized distribution and portability of virtual appliances.
- It aims to support robust installation. Compatibility with the local virtual hardware ia also verified.
- It supports both single and multi-virtual machine configurations. With OVF, Software Developers can configure complex multi-tiered services consisting of multiple interdependent virtual appliances.
- It enables portable VM packaging. OVF is virtualization platform independent.
- It supports a wide range of virtual hard disk formats used for virtual machines today, and is extensible to deal with future formats that are developed.

It supports vendor and platform independence as it does not rely on the use of a specific host platform, virtualization platform, or host/guest operating system. It is also designed to be extended as the industry moves forward with virtual appliance technology. **Usage:** VMDK is a popular file format that encodes a single virtual disk from a virtual machine. However, a VMDK does not contain information about the virtual hardware of a machine, like CPU, memory, disk, and network information, making manual configuration costly. OVF provides a complete specification of a virtual machine. This includes the full list of required virtual disks plus the required virtual hardware configuration, including CPU, memory, networking, and storage. An administrator can quickly provision a virtual machine into virtual infrastructures with little or no manual intervention. **Analysis and Recommendation:** OVF is a portable format that allows users to deploy virtual machines in any hypervisor that supports OVF. As such, it sits at the core of resource management in the infrastructure provisioning layer, overcoming previous deficiencies in standardised solutions such as VMDK. Despite, supporting interoperability as a standard for this specific technical context, other features of OVF are important for cloud architecture. For instance, the localisation support is important for cloud services to be offered across different locales. If these locales can be defined and adapated supported by standards, a hurdle for exploitation is overcome.

D. Security Concerns

Authentication and identity management is a primary concerns for controlling access to cloud resources. Thus, OAuth shall be covered in a brief overview of security concerns. SCAP is a protocol to deal with downloading security content. OAuth as an identity management and SCAP as a security content related standard are covered in the security context.

OAuth is an open standard for authorization. It allows users to share resources across sites. (e.g. photos, videos, contact lists) stored on one site with another site without having to hand out their credentials, typically supplying username and password tokens instead. OAuth uses username and password tokens. A token grants access to a specific site for specific resources and for a specified duration. OAuth runs on top of HTTP or HTTPS. The OAuth mechanism allows users to grant a third party access to their resources (information) stored with another service provider (which could be a cloud provider), but without sharing access permissions. Twitter is one of the users of OAuth, as is Facebook. OAuth is a service complementary to other identity management mechanisms such as OpenID.

SCAP is the Security Content Automation Protocol. It is a protocol to enable automated vulnerability management, measurement, and policy compliance evaluation. It actually combines a number of open standards that deal with software flaws and configuration issues related to security. NIST is in charge of SCAP. SCAP validation focuses on evaluating versions of vendor products, based on the platforms they support. Validation certificates will be awarded on a platform-by-platform basis for the version of the product that was validated. As it attempts to standardize the automation of the linkage between computer security configurations, it is interesting from a cloud interoperability perspective. Trustworthy cloud systems is the aim within which SCAP can be applied. SCAP provides tools that can, e.g., help determine compliance of security requirements implemented in Cloud provider OS images.

IV. Discussion and Conclusions

Interoperability between clouds, cloud services and components is vital for the further development of the cloud ecosystem and market. While standards for the Web Services context are abundant, more specific standards for the cloud computing domain reflect the current maturity. Firstly, a number of standards for the lower infrastructure layers apply to respective cloud computing technologies. They address interoperability solutions for specific aspects like virtual machine management or data management. It reflects initiatives for interoperability for large offerings provided by multinational organisations. Secondly, for platforms and services, the respective (Web) service standards are still of relevance. Standards exist, beyond the core Web services platform, that can further the development of platform and software services from existing offers. Generic service solution can provide a starting point where cloud-specific standards are lacking. This indicates more development in the second category. In addition, it is worth looking at a number of different concerns that help us to judge the state of standardisation and it's impact on interoperability: (i) organisations behind standards and their domain, (ii) stakeholders involved through standards and (iii) standards and open-source/proprietary solutions.

Firstly, by looking at the organisations behind the standards, we can also observe that while the Web services domain is primarily dominated by W3C and OASIS in terms of standardisation, the situation in cloud computing is more diverse. Some of the organisations active include DMTF (management of distributed IT systems), the OGF (grid computing), the OMG (middleware), SNIA (storage), OASIS (services), OCC (cloud), as well as national (e.g. NIST) and sector-specific (e.g. ETSI - telecoms) organisations. Currently, there is a dominance of infrastructure and lower-level management, i.e. enabling concerns for cloud computing, reflecting predictions made in reports such as the EU report on cloud computing and its development time lines [3].

Secondly, stakeholders are yet another perspective that we can look at. We have referred to stakeholders in the review and discussion of standards where relevant to differentiate the different interoperability needs of stakeholders in clouds as multi-organisational, multi-role environments. While the infrastructure standards target clearly software developers, the more generic service-oriented standards are more at the interface (as-a-service) level, targeting service providers and consumers. Particularly combined roles, such as prosumers or aggregators that are providers and consumers by combing and brokering between more basic offerings and somehow extended or advanced needs of end-users, benefit from the recent service description and modelling standards.

Thirdly, while standards can achieve interoperability, often de-facto standards emerge from open-source or proprietary solutions. We dicussed OCCI and CIMI as standards in a context where OpenStack is a strong open-source framework, all competing with Amazon EC2 as the dominant solution.

Our observations do not reflect to a full extent concerns raised by actual and potential cloud users, such as security, privacy and trust [1], but rather indicate more technology concerns in relation to development and deployment activities. By looking at the standards we reviewed here for indications of future standardisation needs, emerging from the categorisation of standards are the following observations:

- Modelling under incorporation of a variety of standards can support migration and, consequently, the uptake of cloud computing solutions.
- Composition, e.g. mashups, is becoming of importance to provide a market for basic and composite offering where providers and aggregators compete.
- Quality of Service and Service Level Agreement standardisations beyond security concerns in the cloud are actually largely lacking.

Open-SCA and other standards in this context are examples of the emergence of programming and interoperation models for services, which will be instrumental for the composition and customisation of cloud services. Adding more semantics to service descriptions is a direction that can further the composition and brokerage in cloud architectures. Interoperability is, once platform stability has been reached, of increasing concern. Migration and interoperability for service offerings are considered for instance in modelling frameworks such as USDL. The need to support composition, brokerage and mediation is also reflected by EMML, which addresses mashups.

ACKNOWLEDGMENT

A number of research project, particularly the EU FP7 projects SLA@SOA, 4CaaSt and Remics, provided invaluable input for this investigation.

REFERENCES

[1] 451 Group. *Report on Cloud Computing 'As-a-service' market sizing* - Report II. 2010.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica. A view of cloud computing. *Communications of the ACM*, 53(4):50–58. 2010.

[3] EU Commission. *Report on The Future of Cloud Computing - Opportunities for European Cloud Computing Beyond 2010*. EU. 2010.

[4] K. Boukadi, C. Ghedira, S. Chaari, L. Vincent and E. Bataineh. How to employ context, web service, and community in enterprise collaboration. *Proceedings of the 8th Intl Conference on New Technologies in Distributed Systems*. ACM, 1-12. 2009.

[5] R. Buyya, J. Broberg, and A. Goscinski. *Cloud Computing - Principles and Paradigms*. Wiley. 2011.

[6] P. Fingar. Cloud computing and the promise of on-demand business innovation. *Intelligent enterprise*. 2009.

[7] C. Pahl, S. Giesecke and W. Hasselbring. An Ontology-based Approach for Modelling Architectural Styles. *European Conference on Software Architecture ECSA 2007*. Springer. 2007.

[8] M.X. Wang, K.Y. Bandara and C. Pahl. Integrated constraint violation handling for dynamic service composition. *IEEE Intl Conf on Services Computing*. pp. 168-175. 2009.

[9] DMTF Distributed Management Task Force: Interoperable Clouds. http://www.dmtf.org/sites/default/files/ standards/documents/DSP-IS0101_1.0.0.pdf. Accessed April 2013.

[10] GICTF Global Inter-Cloud Technology Forum: Use cases and functional requirements for inter-cloud computing. http://www.gictf.jp/doc/GICTF_Whitepaper_20100809.pdf. Accessed April 2013.

[11] OMG Object Management Group: Cloud Interoperability Roadmaps Session. http://www.omg.org/news/meetings/tc/ca/special-events/Cloud_Interop_Roadmaps.htm. Accessed April 2013.

[12] CloudCom 2011 Workshop: Market Implementation of Cloud Interoperability and Portability Research in IaaS and PaaS. http://www.cloud4soa.eu/workshop2011. Accessed April 2013.

[13] Cloud Standards Overview. http://cloud-standards.org/wiki/index.php?title=Main_Page. Accessed April 2013.

[14] DMTF istributed Management Task Force. Open Virtualization Format Specification Version 1.0.0. http://www.dmtf.org/standards/published_documents/ DSP0243_1.0.0.pdf. Accessed April 2013.

[15] Memcached Project web site. http://memcached.org/. Accessed April 2013.

[16] OVF Open Virtualization Format. http://www.dmtf.org/sites/default/files/ standards/documents/DSP0243_1.1.0.pdf. Accessed April 2013.

[17] VEPA Virtual Ethernet Port Aggregator. http://www.ieee802.org/1/files/public/docs2008/new-congdon-vepa-1108-v01.pdf. Accessed April 2013.

[18] OCCI Open Cloud Computing Interface. http://occi-wg.org/. Accessed April 2013.

[19] USDL Unified Service Description Language. http://www.w3.org/2005/Incubator/usdl/. Accessed April 2013.

[20] EMML Enterprise Mashup Markup Language. http://www.openmashup.org/omadocs/v1.0/index.html. Accessed April 2013.

[21] Open CSA - Open Composite Services Architecture. http://www.oasis-opencsa.org/. Accessed April 2013.