2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing

Identifying Contextual Properties of Software Architecture in Cloud Computing

Khaled M. Khan
Qutaibah Malluhi
Department of Computer Science and Engineering
Qatar University, Qatar.
email: {k.khan, qmalluhi}@qu.edu.qa

***Abstract*—This paper argues that the contextual properties of cloud-enabled software architecture should be identified and understood differently by cloud consumers. The existing architectures are not designed to exploit the contextual properties of cloud computing. The emergence of cloud computing virtually forces cloud consumers to re-evaluate their software architectures in light of the cloud computing context which requires relinquish control over most architectural components to cloud providers. In a cloud-enabled architecture, the shift of ownership of and control over architectural components from consumers to cloud providers has profound impact on the ways cloud consumers design their software architecture. In this perspective, we go beyond the traditional definition of software architecture, and introduce the concepts of architectural scope, ownership, and control as the contextual properties of an architecture.**

***Keywords*-software architecture, control over architectural components, architectural scope, ownership, contextual properties, cloud-enabled environment.**

I. Introduction

Cloud computation is a paradigm shift to a new service based computing similar to the notion of one-stop-services. It presents a range of simplified services that are flexible enough for consumers to choose from. The main idea behind cloud computing is to provide *x-as-a-service* to organizations, meaning *x* could be substituted by any dynamically scalable service such as software, database, security, hardware, platform, storage etc. Instead of installing application software in their machines, buying powerful computers and managing servers, organizations can use application software as well as computers owned and managed by someone else. Consumers need not have much expertise in or control over the computing resources that support the services they consume.

In spite of the obvious benefits offered by cloud computing, how organizations will define their software architecture in cloud-enabled environment is a challenge for the propelling cloud computing adoption in a wider scale. The software architecture of an organization has an impact on how it manages and controls its computing needs and resources in cloud settings. In current practices, software architecture consists of several architectural components that are mostly owned and controlled by the organization that uses it, and are limited within the boundary of that organization. Organizations currently have full control over and ownership of their architectural components. The cloud computing is currently in the process of changing this practice. This change has impact on how computing resources are shared, operated, used, and managed by organizations.

The migration from localized computing environment to external services presents challenges to software architects. The challenges include defining contextual properties of architectural components, finding their impact, understanding the controlling aspects of architectural components, and making balance between consumer-controlled versus cloud-controlled architecture. These issues force software architects to re-consider the existing practices in light of the new frontier of architectural challenges previously unaccounted for. Although the contextual properties have no direct observable manifestation in a software architecture, these do point out something important for all cloud computing stakeholders.

In cloud computing, the scope of software architecture stretches from the organizational to cloud provider's boundaries. It is not only the *architectural scope* that has changed, the *control* over architectural components has shifted to cloud provider too. The *ownership* of architectural components has also transferred from organizations to cloud providers. This changing landscape has impact on the success of cloud integration and service maintenance. Cloud consumers need to create a service integration model to help implement the service evolution and maintenance successfully [10]. Without a thorough understanding of the contextual properties in cloud based environments, such integration is difficult to achieve.

Cloud computing has disrupted the architectural landscape that organizations have so far enjoyed in designing and managing to meet their computing requirements. The limited understanding of the underlying contextual properties of software architecture prevents many organizations from identifying the real impact of cloud-enabled services. Existing practices of software architecture are either too IT product-specific or too shallow. A more deeper understanding in cloud-based architecture is required to enable organizations for a wider adoption of this paradigm. The interplay

978-0-7695-4612-4/11 $26.00 © 2011 IEEE
DOI 10.1109/DASC.2011.104

between the software architecture of an organization and the new characteristics of cloud computing should be well balanced and understood by all stakeholders [7]. In this paper, although we primarily focus on software architecture in order to identify its contextual properties in cloud computing, we also include the infrastructure components (hardware devices and system software) of the system in our discussion.

There is no systematic treatment yet of contextual properties of software architecture in relation to the paradigm shift to cloud computing. The topic is under-represented in research forums. Since contemporary research has not focused much on these properties, this paper attempts to fill this gap. Some architectural styles are proposed, in particular for multi-tenant systems such as REST [4] and SPIAR [9]. REST focuses on the communicated data elements, whereas, SPIAR emphasizes on interactive user interface and on interaction between client and server. Recently, a new architectural style called SPOSAD [8] for multi-tenant software applications has been proposed. It focuses on architectural constraints, and discussed various trade-offs to consider by the architect. A cloud computing open architecture model comprising seven principles has been proposed in [13]. The main objective of the model is to get cloud vendors, cloud partners and cloud consumers to work together based on the seven principles.

In this paper, we apply software architecture concepts and methodologies to cloud computing in order to understand the contextual properties and their implications on the design decisions and inevitable trade-offs. Cloud computing demands from software architects to rethink about the new landscape of software architecture that encompasses from organizational boundaries to cloud providers' perimeters. We should understand how architectural properties change for the dynamically instantiated, distributed and ephemeral services [1]. The line of thinking presented in this paper is expected to result in the following benefits:

- *First*, organizations can map which architectural components belong to them and which are to cloud providers. This will enable organizations to distribute IT resources and the workload efficiently to its IT staff. The allocation of resources to specific architectural components could be done more optimal way. It would, hence, allow organizations to manage their IT resources better.
- *Second*, service level agreements (SLAs) could include a clear mapping of the ownership of architectural components between consumers and cloud providers. Based on the clear distribution of architectural components and their control flows, billing could be done effectively and accurately. The demarcation of responsibility could be well defined based on the architectural properties mapped onto the cloud-enabled architecture.
- *Third*, the thorough knowledge in contextual properties of software architecture would set the right context in which the cloud consumer could ask the cloud provider the right questions about the desired behaviors of the system. The consumer is in better position to know which functionality they could customize, which functionalities could be modified by whom (consumer or cloud provider) in order to add more capabilities. Hence, software maintenance and evolution of software architecture could be well managed.
- *Fourth*, the distribution of ownership, control and architectural scope across organizations would provide enough information to the consumer to define their own security and privacy policy around different architectural properties. The consumer could find where they need to put their own security shield. They can also know which components of the architecture are under the security policy of the provider. This would help organizations to achieve an optimal interoperability of their security policies across organizational boundaries.
- *Finally*, the presence or absence of one or more architectural elements under their control would help the cloud consumer to bind the problem space as well as the solution space in their cloud based architecture. It would provide organizations with better control on their architecture as well as technology resources.

Our main goal of this paper is to flesh out contextual properties of software architecture that are vital for all stakeholders to understand in a cloud based environment. The identified properties are expected to contribute to the understanding of software architecture on cloud based systems. In the reminder of the paper, section II introduces the basics of software architecture with an example. In section III, contextual properties are discussed in relation to cloud based software architecture. Cloud computing and the associated architectural issues are discussed along with the major cloud service models in section IV. Section V presents cloud-enabled architecture, and shows how various contextual properties of architectural components have changed. An example to illustrate the differences between the localized software architecture and cloud-enabled architecture in terms of scope, ownership and control is provided in section VI. A discussion is presented in section VII. Section VIII concludes the paper.

II. Software architecture

A software architecture is a conceptual model that defines architectural components from which a system is constructed using rules of interactions and connectors [12]. More precisely, a software architecture is built from the following elements [2]:

- *Architectural components*: A set of architectural components that perform some predefined functions, or represent specific information. These are either the

computational units with well-defined interfaces such as modules, procedures, processes, filters; or pieces of information such as objects, data, databases, etc.

- *Connectors*: These are the compositional mechanisms for gluing the components together in a topology. Examples of connectors include procedure calls, shared variables, data flows, messages, communication protocols, events, pipe streams etc.
- *Architectural styles*: A set of rules that mediate communication, coordination, and interaction among architectural components. The choice of interactions is usually guided by the rules of an architectural style, and the interaction forms the geometric layout and control flow of the software system. In other words, an architectural style defines a vocabulary of architectural components, their interactions, the control flow among components [11]. Examples of styles are pipe-and-filter, process model, object-oriented, etc. Different styles have different structure and topology.

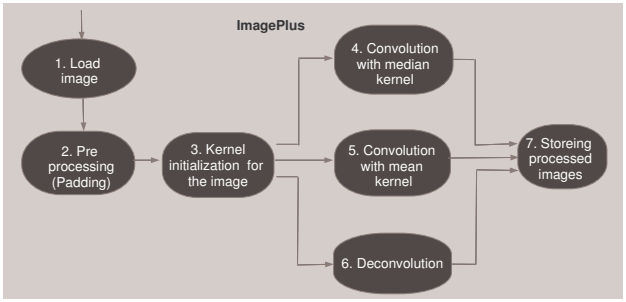


Figure 1. Software architecture of the image processing system.

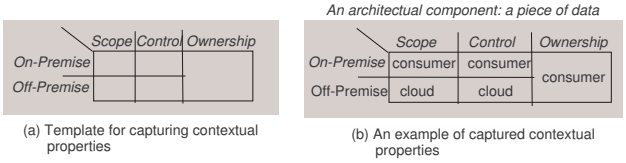
To set the right context and illustrate the above elements, consider an example of a system. A company called *ImagePlus* processes digital images. The company is specialized in image smoothing, removing blur from the image, and detecting edges in the image. The process heavily relies on the convolution and deconvolution operations. Convolution involves shifting the *kernel* (another image with less number of pixels) over the original image, multiplying the kernel's pixel values with the corresponding pixel values of the original image, and take the sum of these multiplications. The process uses different types of kernel such as mean kernels, median kernels etc. Deconvolution is used to remove blur from a blurred image due to out-of-focus or motion while capturing the image with a camera.

Figure 1 provides a snapshot of the software architecture that uses pipes-and-filters architectural style. Filters (dark oval) are the architectural components, and pipes (arrows) are the connectors. The numbering of the filters denotes the sequence of the process. The rules of interaction in this style are: (i) each filter (component) reads streams of data on its input, and produces streams on its output channels; (ii) the pipes (connector) serve as conduits for the streams, transmitting outputs of one filter to inputs of another; and (iii) filters are independent and do not share state with other filters. The shaded rectangle in the background denotes the scope of the architecture in terms of organizational boundary of ImagePlus. The system is currently running on Linux platform using three HP ProLiant ML 350 G6 servers, and processed images are stored in a Redundant Array of Independent Disk (RAID). All hardware devices belong to ImagePlus.

III. Contextual properties

Cloud based services consumed by organizations usually require different ways of integrating architectural components, different design issues, and different kinds of reasoning depending on the operational environment. Although the business goal remains same, the contextual properties of software architecture change due to the new operational environment. We identify the contextual properties of software architecture which were previously unaccounted for. In this example, we can define three contextual properties: *architectural scope*, *ownership*, and *control*.

In a nutshell, a software component has a scope, it may belong to a single entity, and it must be controlled by an entity at different *stages* of the service. In cloud computing, an architectural component may be in one or both of these two stages: *on-premise* and *off-premise*. The stage of an architectural component is said to be *on-premise* while it is within the boundary of the consumer's organization, whereas, *off-premise* denotes the stage of the component while it is in the boundary of the cloud provider. Figure 2(a) shows a template to capture the contextual properties of an architectural component such as a process, an object, data, server etc. Figure 2(b) depicts an example showing that while data in on-premise of the consumer, the scope and control over the data are held by the consumer who is the owner of the data. However, the same data is in the scope of and under the control of the provider although the ownership remains unchanged while it is off-premise of the consumer. We now discuss each of the properties.



(a) Template for capturing contextual properties

(b) An example of captured contextual properties

Figure 2. Capturing contextual properties of architectural component

A. Architectural scope

An architectural component can be located in the boundary of either the consumer or cloud provider in either

on-premise or off-premise stages respectively. Architectural scope refers to the physical distribution or geo-location of the architecture and its components. The topology of an architecture may spread across several distributed locations beyond the consumer organization at one of two different stages (on-premise and/or off-premise). That is, some of the architectural components may be located outside of the consumer's organizational boundary. For example, a process of a software architecture may run on a server that belongs to another organization. Similarly, a server in the hardware architecture of an organization may be installed in another organization, the disks may be located in a third organization.

The example architecture (Figure 1) of the image processing system is located in a single organization, that is, it is located within the perimeter of ImagePlus. Components in motion, such as a piece of data, can have two stages in two different times. A piece of data can travel from the consumer's organization to the provider's boundary. In that case, it can be in two different stages -once on-premise and then off-premise while it is processed or stored on cloud. However, a process or a hardware device can have only one stage in its lifetime because these components are static. The scope can also be related to another contextual property –the *ownership*.

B. Ownership

Ownership means the possession of architectural components along with the right to alienate that possession. More specifically, the owner possesses an entire architecture or some of its components is different stages. Ownership does not change stage from on-premise to off-premise or vice versa. Figure 2 shows that it is not possible for the data to have two different ownerships in two different stages. The ownership of a component is fixed across organizational boundaries. An organization may use a software architecture or some of its components which are owned by another organization. In our example, all filters, pipes, and devices are owned by ImagePlus. The company does not share the software components as well the hardware with any other organizations.

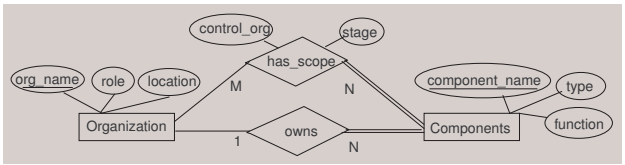
In majority cases, the *ownership* is related to the *scope* of the architecture. If an architecture or its components are located within the scope of an entity, most likely these are owned by that entity. However, there are instances where components are localized within the perimeter of a company, but the company does not own it. For example, a rented server along with an application may be located on-premise of the renting entity, but these are owned by the lending organization. Ownership has an interesting relationship with another contextual property, called *control*.

C. Control

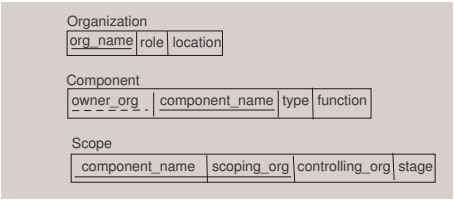
Controls mean the right to make design decision on the architecture and the architectural components. It is the ability of the organization to make system related design decisions that affect the operational as well as structural aspects of the architecture. The controller of an architecture (or some of its components) makes the system related decisions such as maintenance, evolution, fixing bugs, removing and adding components. The control over a static architectural component cannot change at different stages of the service from the on-premise to off-premise, or vice versa. The control is fixed because static components do not move within the architecture. However, the control over data-in-motion can change from on-premise to off-premise, or vice versa as it travels from one boundary to another.

Control is usually coupled with *ownership*, and some cases with the *scope* of architectural components. That is, if an organization owns an architecture or its components, it controls that entity. It is also related to the scope of the architecture, that is, the boundary of the architecture may determine who controls it. However, this is not the case for the rented or outsourced architectural components because although these are located within the boundary of an entity, but that entity does not control them. The lending organization controls these components. In our example, ImagePlus controls all architectural components, connectors and infrastructure devices.

A modeling tool such as an E-R diagram can capture



a) E-R diagram



b) Relational schema

Figure 3. A conceptual model and its mapping to schemas.

these contextual properties of a software architecture. The model can be used to develop relational schema to store contextual properties such as which organization owns which component, their scope and control issues. Figure 3 depicts an example of such conceptual model and the corresponding schemas. According to the diagram in the figure, an architectural component can only be owned by one organization,

however, a component could be located in more than one architectural boundary and controlled by more than one organization in different stages.

The software architecture community never looks software architecture in light of these contextual properties because there was no need for these due to uni-ownership, uni-location, and uni-control of the software architecture. Most software architectures are localized within the organizational boundary. In a cloud based environment, it is interesting to see how the above contextual properties of software architecture are distributed, have an impact on cloud-enabled systems, how the architectural styles evolve to address the cloud characteristics, and how to balance the control over and ownership of various architectural components. We explore some of these with an example in the rest of the paper.

IV. Cloud service models

It is vital for the organization to know how control is shared, allocated, transferred among components, how organizational data interact with cloud-controlled processes in a cloud-enabled software architecture, and how the ownership affects the design decision of the system. In order to understand these perspectives, we first need to explore the properties of major cloud service models. Cloud computing has three major service models: infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS). Each of these is briefly discussed.

- *Infrastructure as a service (IaaS)* provides computer infrastructure -typically hardware devices -as a service, such as storage, CPU and networking. Cloud consumers use these hardware devices as a fully outsourced service instead of buying and managing them. The hardware devices owned by cloud provider in the form of either a virtual machine instance or storage. Once created, the consumer can load any operating system and application software they choose on the virtual machine instance, and put their data on the storage.
- In *Platform as a service (PaaS)*, the consumer uses the platform provided by the cloud provider. The platform manages the deployment and availability of the consumer's application software. It comes with the underlying hardware, the operating system, and the development tools that support software development. Examples of such resources are the development platform, software components, design tools, compilers, and testing suites. Consumers of PaaS can write their applications directly to the rented platform, and elastically scale out as needed [6].
- *Software as a service (SaaS)* completely removes the need for the customer to be concerned with anything such as hardware, platform and software, but application capability. The main resource-based architectural component of the SaaS is the application resources that support services accessible to cloud computing consumers. Virtualization is the key architectural component in cloud computing for creating and managing the cloud-specific virtual machines.

In a nutshell, IaaS provides hardware devices, PaaS offers platform which automatically comes with hardware devices and the operating environment. SaaS includes the application software with the underlying hardware devices and platform. Virtualization technology is the core for enabling the cloud resource sharing among multiple consumers. Virtualization manages the pro-creation and allocation of hardware and software virtualization to the right physical machines of the consumer [13].

V. Cloud-enabled architecture

A cloud-enabled architecture may comprise the consumer's on-premise as well as cloud resources such as services, middleware, software components, geo-location, and the interactions of components [5]. This paper focuses on the relationship between the contextual properties and three major cloud service models. The control over architectural components is the most important contextual property in cloud computing. In cloud architecture, the cloud provider has their own autonomous control flow of the system. In a non-cloud environment, the software systems and hardware devices are localized in the organization that uses it. Organizations have unified control flow in the architecture tuned to customize their organizational needs. These two practices are interwoven in a cloud based system architecture. In cloud computing, organizations should restructure their existing architecture to adaptive computing services offered by cloud computing.

Figure 4 shows the division of high-level architectural components of different service models based on control. The architectural components (infrastructure devices, software) are grouped into two distinct types in three cloud service models: *client-controlled components* and *cloud-controlled components*, represented as ovals and rectangles

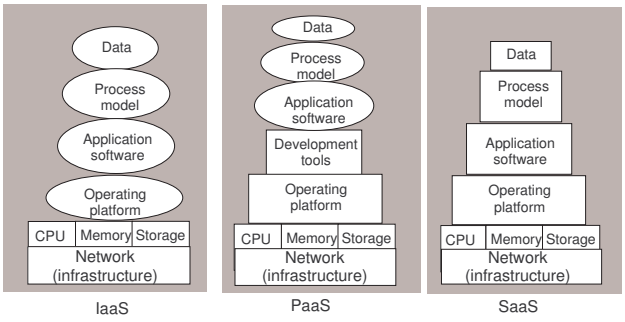


Figure 4. High-level architectural components of service models, and the state of control

respectively. The former are those that are controlled by cloud computing consumers, namely organizations, whereas, the latter components are controlled and managed by cloud providers, both in the off-premise stage. Note that data, process model, application software, operating system are the components of software architecture, whereas CPUs, memory, storage and network are the components of infrastructure.

In the *IaaS* model, the consumer-controlled architectural components typically are data, process model, application software and operating platform. The provider controls the hardware devices such as memory, disks, machines, network etc. However, this may change from application to application. If the consumer receives software services in addition to infrastructure, the control over most components is held by the provider. In PaaS, the cloud provider controls hardware infrastructure, operating platform and the development tools, whereas, the cloud consumer controls application software, process and data. In SaaS, the consumer virtually relinquishes their control over all architectural components to the cloud provider.

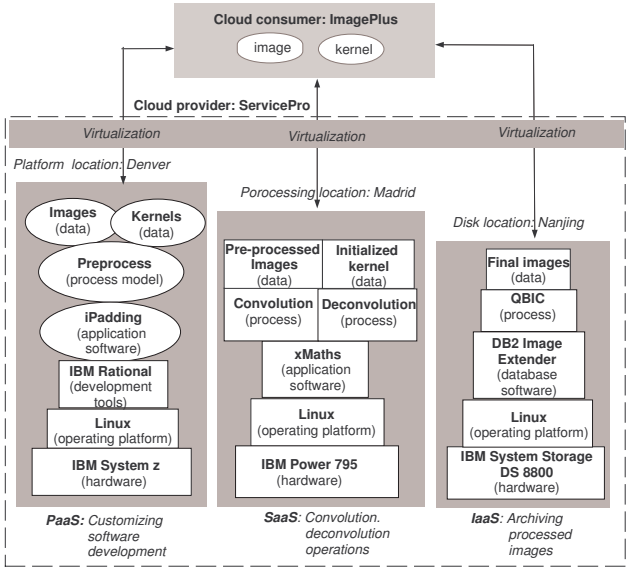


Figure 5. Image processing architecture consigned on Cloud

VI. AN EXAMPLE

ImagePlus has decided to outsource its image processing software and hardware to a cloud provider called *ServicePro* in order to maximize the competitive advantage by reducing cost. The ServicePro uses object oriented architectural style to process the convolution/deconvolution operations. ImagePlus also redesigns its pre-processing and kernel initialization components in object oriented to make the architecture compliance with the provider. Our example image processing architecture is now mounted on a cloud based environment as shown in Figure 5. ImagePlus plans to use the following 'pay-as-you-go' cloud service models with ServicePro:

- *A PaaS for the customizing its image padding and initializing processes*. ImagePlus develops and customizes its pre-processing software for padding the edges of the original image. The image is first zero-padded or row-replicated to include additional rows and columns at the edges. This pre-processing is done in accordance with the size of the kernel being selected. ServicePro provides ImagePlus with the IBM System z machine, and Linux operating platform along with the IBM Rational Development tools including compiler, debugging tools etc (a PaaS). ImagePlus develops a customized application software called *iPadding* using IBM Rational development tools. The output of the pre-processing and initialization are the padded original image and the initialized kernel respectively. These two objects together referred to 'images' (data) in this paper are sent to the application software running on the cloud (a SaaS). See Figure 5. Note that an oval denotes a consumer-controlled, and a rectangle means a cloud-controlled architectural component.

Table I summarizes the scope, ownership, and control

Table I
SCOPE, OWNERSHIP AND CONTROL DISTRIBUTION IN PAAS

	Component	Scope	Owner	Control	Stage
1	IBM Sys z	ServicePro	ServicePro	ServicePro	OFF
2	Linux O/Sys	ServicePro	ServicePro	ServicePro	OFF
3	IBM Rational	ServicePro	ServicePro	ServicePro	OFF
4	iPadding	ServicePro	ImagePlus	ImagePlus	OFF
5	Preprocess	ServicePro	ImagePlus	ImagePlus	OFF
6	Image	ImagePlus	ImagePlus	ImagePlus	ON
7	Image	ServicePro	ImagePlus	ImagePlus	OFF
8	Kernel	ImagePlus	ImagePlus	ImagePlus	ON
9	Kernel	ServicePro	ImagePlus	ImagePlus	OFF

distribution of the architectural components (infrastructure, software) along with the stages (on-premise and off-premise) in PaaS. In the table, ON means on-premise and OFF denotes off-premise. The table shows some interesting properties of PaaS in this example. The customized software iPadding (developed by ImagePlus) is located in ServicePro (scope), but it is owned and controlled by ImagePlus in the *off-premise* stage (see row 4). Similarly, the process model and data (image and kernel) are owned and controlled by ImagePlus but located in two different boundaries in two different stages of the service (rows 6 & 8 on-premise, and rows 7 & 9 off-premise). The images and kernels are within the boundary of ImagePlus before processed (on-premise stage) by the software iPadding (see rows 6 & 8). These objects are in the perimeter of ServicePro while processed (off-premise

stage) by iPadding (see rows 7 & 9). Notice that the control remains with ImagePlus, although the scope has changed from on-premise to off-premise (rows 7 & 9). Images are pre-processed, and kernels are initialized by the software owned by ImagePlus.

- *A SaaS for the convolution and deconvolution operations for digital images*. According to the SLAs, ServicePro provides an IBM Power 795 computer, xMaths software, and the Linux operating platform for the operations. ImagePlus will only supply pre-processed images and the initialized kernel for the operations. These two input data are fed by the customized software *iPadding* (developed on the PaaS) to *xMaths*. Table II summarizes the scope, ownership, and control distribution of the architectural components (hardware, software) along with two stages in SaaS. Notice the last four rows of the table. ImagePlus owns the images and kernels, but these are located in two different perimeters in two different stages, namely, rows 5 & 7 (on-premise stage), and rows 6 & 8 (off-premise stage) respectively. Similarly, the control over images and kernels has changed according to the architectural scope. ServicePro gets the control over these two objects when these are processed by its machines and software within its boundary (see rows 6 & 8). ImagePro does not have much control over any components including its data while on the cloud.

Table II
SCOPE, OWNERSHIP AND CONTROL DISTRIBUTION IN SAAS

	Component	Scope	Owner	Control	Stage
1	IBM Power 795	ServicePro	ServicePro	ServicePro	OFF
2	Linux O/Sys	ServicePro	ServicePro	ServicePro	OFF
3	xMaths	ServicePro	ServicePro	ServicePro	OFF
4	Convolution Deconvolution	ServicePro	ServicePro	ServicePro	OFF
5	Image	ImagePlus	ImagePlus	ImagePlus	ON
6	Image	ServicePro	ImagePlus	ServicePro	OFF
7	Kernel	ImagePlus	ImagePlus	ImagePlus	ON
8	Kernel	ServicePro	ImagePlus	ServicePro	OFF

- *An IaaS for the archiving images electronically*. ImagePlus also uses huge data storage capacity such as IBM System Storage DS8000 services provided by ServicePro in order to store its digital images which are processed and archived by the software xMaths (SaaS) of ServicePro. The IaaS also comes with the database software DB2 Image Extender along with the query processor QBIC. Table III summarizes the scope, ownership, and control distribution of the architectural components (hardware, software) in IaaS. The table shows that ImagePlus owns the final images, although these are stored in the boundary of the provider (last row). However, the control on the stored images is held by ServicePro.

Table III
SCOPE, OWNERSHIP AND CONTROL DISTRIBUTION IN IAAS

	Component	Scope	Owner	Control	Stage
1	IBM Storage DS8800	ServicePro	ServicePro	ServicePro	OFF
2	Linux O/Sys	ServicePro	ServicePro	ServicePro	OFF
3	DB2 Image Extender	ServicePro	ServicePro	ServicePro	OFF
4	QBIC Process	ServicePro	ServicePro	ServicePro	OFF
5	Final Image	ServicePro	ImagePlus	ServicePro	OFF

VII. DISCUSSION

Figure 5 and three tables suggest that all key architectural components of ImagePlus are spread across various locations. Most of these now belong to cloud computing except the images (data) and the customized software (for the pre processing) developed on PaaS. A further details of the models reveal that different processes, disks and hardware devices used in these services are physically distributed at various locations (see Figure 5). The PaaS is provided by the Denver (USA) cloud location of ServicePro; the convolution and deconvolution operations in SaaS are served by the software running on computers located in Madrid (Spain); the convoluted/deconvoluted images are stored in disks located in Nanjing (China). Without the contextual properties along with the entire cloud based software architecture, ImagePlus may not be aware of these various geo-locations unless explicitly informed by ServicePro.

The example demonstrates that with the contextual properties, cloud consumers could clearly identify which architectural components they own and which they don't. Although ownership is always associated with control, but this is not always the case as we have seen in the example. It is clear that in a cloud-enabled environment, consumers give up much of their control of their data and processes due to changing contextual properties in a cloud-based architecture. Once the consumer's data leave the organizational boundary, control is held by the cloud service provider in most cases. In our example, since ImagePlus has decided to consign all digital images to ServicePro, it does not have much control over the processes and images while they are processed and stored at the cloud sites of ServicePro. ImagePlus has lost control over majority of its architectural components that are now being designed, developed, owned, and managed by ServicePro.

These changes have also other implications. The cloud consumer also loses flexibility regarding the extension or modification of processes as its control diminishes. In the example, ImagePlus has lost some of its flexibility on the processing of images. It now heavily depends on ServicePro to do the convolution/deconvolution operations.

Another concern resulted in from the diminishing control, ownership and scope is the security and privacy. ImagePlus

may have valid reasons to worry about the confidentiality and privacy of their images now stored in devices owned and managed by ServicePro because the images are persistently stored in the databases and storages owned and controlled by ServicePro.

With the mapping of the architectural components on the cloud based architecture along with the contextual properties, the cloud consumer could find more about their degree of control over the architectural components, the ownership along with the responsibilities and the architectural scope. These would enable them to plan their resources, design software architecture addressing their business needs and monitor the usefulness of the cloud based architecture. As Grady Booch [3] rightly states in a different context, an architecture is a declaration of the shared reality that represents a common vision among a set of stakeholders, and represented by a set of interlocking models. This is quite true for a cloud based architecture where both the consumer and the cloud provider share their understanding of the system in interlocking artifacts. Software architecture associated with the contextual properties could enhance the interlocking, and advance the understanding of the artifacts.

VIII. Conclusion

The new way of service consumption in cloud computing requires the contextual properties of software architecture to be defined in terms of its key architectural components, their interactions, the control flow, ownership of components, topological distribution of components, geo-location of architectural components. The contextual properties of software architecture with clear control flow provide an organizational alignment to everyone in the organization to understand the impact of outsourcing computing needs to cloud. An architecture in a cloud-enabled environment is expected to show how and where cloud computing services fit into the organizational IT strategy, and how it has impact on the way cloud computing delivers services to the business goals of the organization.

Transplanting a traditionally-structured architecture onto a cloud platform without right contextual properties would mean trying to fit a square object into a round hole. The architecture should be designed such a way that it is easily understandable and manageable across the organizations. A balanced need is to be struck in developing architectures that can be used to provide enough technical information as well as contextual properties to all stakeholders. Software architecture might be behind the scenes, but it is highly relevant to cloud consumers because they could see how satisfying the services are, provided by the underlying architecture and its components.

This study demands further investigation to formalize the contextual properties, and exploration on how to measure the impact of these properties on software architecture. It includes formulating methods that could reason about the presence or absence of these properties in software architecture and their impact on business processes. We acknowledge that this topic requires more deeper analysis in order to flesh out a complete taxonomy of cloud-enabled software architecture and the associated contextual information.

Acknowledgement

This publication was made possible by the support of an NPRP grant from the Qatar National Research Fund. The statements made herein are solely the responsibility of the authors.

References

[1] P. Banarjee and et al., “Everything as a Service: Powering the New Information Economy”, IEEE Computer, Vol. 44(3), March 2011, pp. $36-43$.

[2] L. Bass, P. Clements and R. Kazman: Software Architcture in Practice. Addison-Wesley, 2003.

[3] G. Booch, “Architecture as a Shared Hallucination”, IEEE Software, Jan-Feb. 2010, Vol. 27 (1), pp. $95-96$.

[4] R. Fielding and R. Taylor, “Principled design of themodern Web architecture,” ACM Transactions on Internet Technology, Vol. 2(2), 2002, pp. $115-150$.

[5] G. Kaefer, “Cloud Computing Architecture”, Siemens AG 2010 Corporate Technology, Munich, May 2010.

[6] Y. Khalidi, “Building a Cloud Computing Platform for New Possibilities”, IEEE Computer, Vol. 44(3), March 2011, pp. $29-34$.

[7] K. Khan and N. Gangavarapu,“Addressing Enterprise Architecture in Cloud Computing: Issues and Challenges”, IT Cutter Journal, November 2009, pp. $27-33$.

[8] H. Koziolek, “The SPOSAD Architectural Style for Multi-tenant Software Applications,” Procceings of the Ninth Working IEEE/IFIP Conference on Software Architecture, 2011, pp. $320-327$.

[9] A. Mesbah and A. van Deursen, “A component and pushbased architectural style for AJAX applications,” Journal of System Software, Vol. 81(12), 2008.

[10] M. Papazoglou, V. Andrikopoulos and S. Benbernou, “Managing Evolving Services”, IEEE Software, May/June, 2011, pp. $49-55$.

[11] D. Perry and A. Wolf, “Foundations for the Study of Software Architecture,” ACM SIGSOFT Software Engineering Notes, Vol. 17(4), Oct 1992, pp. $40-52$.

[12] M. Shaw and D. Garlan: Software Architecture: Perspectives on Emerging discipline. Prentice-Hall, 1996.

[13] L. Zhang and Q. Zhou, “CCOA: Cloud Computing Open Architecture”, Proceedings of the IEEE International Conference on Web Services”, 2009, pp. $607-616$.