Cloud providers offer a compelling business model for their customers to outsource the supporting infrastructure or platform needed for a cloud service. In that sense, different service delivery models, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and even Software as a Service (SaaS) are not really as different as they first seem. They must still provide the required hardware and software components (at the appropriate level of abstraction) in order to support the cloud service properly.

However, while the cloud business model may offer cost advantages for the necessary platform and infrastructure, in many cases businesses would be foolish to assume that a cloud provider will be able perpetually to meet a business service owner's requirements.

The reasons are clear. Over time, the business service is likely to evolve and require different infrastructure or platform components with different characteristics, such as scale, resilience, security, privacy, geographic location of stored data, legal jurisdiction, service level agreements, and performance characteristics, to name just a few.

Different characteristics are required because the business services themselves tend to evolve over time. These changes can be substantial, driven by changing business and regulatory requirements, or driven by an ever-evolving customer landscape and market. Or, it may simply be a business relationship change. Perhaps the current cloud provider’s newest, biggest customer is a direct competitor, and there is a concern regarding conflict of interest.

In any case, it is in the nature of things that, over time, a business may find that its strategic services are not properly enabled or well-supported by its current cloud provider. As many comedians and self-proclaimed philosophers have observed, there is a causal relationship between marriage and divorce. A business is well-advised to bear that in mind in its evaluation of cloud providers. In fact, the complexity and cost of porting a cloud service to a different cloud provider (the divorce) is often under-appreciated until too late.

However, despite the clear and urgent need for such portability, until recently, there seemed little hope that the industry could truly move forward to develop the required service portability and lifecycle manageability using a standards-based approach. That is, until now.

The Genesis and Velocity of TOSCA

Industry standards development organizations such as the DMTF and the Open Grid Forum have developed standardized APIs and models, respectively called CIMI\(^1\) and OCCI\(^2\), that enable vendor-agnostic, programmatic control of operational management functions on cloud providers. Similar to proprietary operational management APIs such as Amazon’s EC2 or VMWare’s vCloud, these important, new standard APIs enable clients to perform actions such as requesting virtual machines, network resources, and virtual storage with specific characteristics, often with the ability to specify a specific operating
system or software stack pre-installed. They also provide other capabilities, such as obtaining an invoice for the virtual resources created, as well.

However, deploying and managing a complex cloud service, consisting of many different types of software and hardware, across its entire service lifecycle, from deployment through inevitable reconfiguration and patching needed to meet changing business needs, and ultimately removing that service with the possible necessity of deploying it on another cloud provider with different characteristics is beyond the scope of these two standards, as it requires more than operational management. A higher level of abstraction and control is needed, one that understands the complexities of managing the service lifecycle.

This reality was recognized in 2011 when a consortium of leading cloud vendors and solution providers consisting of CA Technologies, CapGemini, Cisco, Citrix, EMC, IBM, NetApp, PwC, Red Hat, SAP, Software AG, Virtunomic, and WSO2 was formed to jumpstart initial development of a standard. By the fall of 2011, these same vendors had proposed a new Technical Committee (TC) in OASIS. OASIS is a leading not-for-profit standards development organization known for its open process, large membership of public and private sector participants, and proven track record.

The new TC was called the Topology and Orchestration Specification for Cloud Applications TC or TOSCA TC. The TC had its first meeting in December of 2011 and accepted from these vendors a specification they had developed with the intent to further develop the specification in an open, standards development process.

As of this writing in September of 2012, the TOSCA TC has grown to become one of the largest and most active Technical Committees in OASIS, with many additional cloud computing vendors involved. A CA Technologies representative co-chairs the TC. The TOSCA specification is currently on track for completion and public comment, which is required prior to ratification as a standard, by the end of 2012. A Primer consisting of use cases, examples and preliminary TOSCA models based on the TOSCA language are expected to follow shortly after, in the first quarter of 2013.

The TOSCA-Enabled Eco-System, Service Templates, and CSAR

The depth and potential impact of the TOSCA vision can be best understood by looking at similar attempts to deal with the lifecycle of other complex, manmade structures. Skyscrapers, bridges, and other complex structures are constructed, modified, maintained, and even dismantled using industry standard descriptions and manifests in the form of blue prints, materials lists, and operational best practices; some of these best practices are as simple as “build the basement before the roof,” but other best practices are naturally quite complicated and building specific.

In the case of these buildings and other structures, documents and best practices are created, modified, and maintained by an eco-system of architects, designers, real-estate developers, construction crews, and inspectors using various design and tracking tools during the lifespan of the structure. The focus of the documents is primarily at the conceptual level of the building itself, its principal components, construction and maintenance, and not on how those components, like a furnace or elevator, are actually built themselves.

Similarly, TOSCA is designed to support an eco-system based the definition of a
common, machine-readable language for the description, maintenance, and operational management (best practices needed to support cloud services across their lifetime). Think of TOSCA as enabling an eco-system that starts with cloud service developers describing (modeling) the principal components, characteristics, and requirements of a service in a standardized fashion so that the service can be understood, installed (deployed) or removed (undeployed) by many different types of TOSCA-compliant cloud providers with very little additional effort.

In TOSCA, we call these service descriptions or models Service Templates. The focus of the TOSCA Service Template is on cloud service; its principal components, requirements, and characteristics as well as the needed processes to support the cloud service’s lifecycle using the descriptions and best practices encapsulated in a TOSCA Service Template itself.

The TOSCA Topology Template

As of the time that this article is being written, the TOSCA language has largely been defined, although some final details are still being worked out. Nevertheless, the foundational mechanisms are clear. The TOSCA language enables the specification of an XML document called a Service Template. A Service Template is considered instantiated or made “tangible” when the service it represents is deployed. In simple terms, each Service Template consists of a Topology Template that can define the topology (structure) of a service. This topology can function almost like a graph or chart of connected nodes corresponding to various service components and their relationships to each other.

As mentioned above, cloud services are comprised of many types of software components, each of these supported by a virtual and ultimately physical infrastructure; all with their own vast range of attributes, relationships, and requirements. TOSCA provides a standardized language for the description of this topology, including components, their relationship with each other, and the requirements and capabilities of these components.

TOSCA-enabled environments can read and modify this topology, most commonly through the use of optional process models, called “plans,” to define orchestrations using a process modeling language like BPMN. These plans can invoke lifecycle management behaviors (operations) associated with each of the various components of the service in order to achieve operational goals such as deployment, scaling, and patching. In other words, they can “ask” that a component install itself, so to speak. Plans can also be leveraged to encapsulate “best practices.” For example, a Plan could be used to adjust a service deployment depending upon what type of virtual machines are supported by the cloud provider, or based on other capabilities inherit in the cloud provider’s platform. Plans and the various related scripts and other files relevant to TOSCA are considered Installation Artifacts in TOSCA, as opposed to Deployment Artifacts such as application TAR files, etc.

TOSCA Node Templates enable the specification of components such as databases, application servers, operating systems, as well as virtual machines, storage, and even physical devices. Each Node Template in the service represents an occurrence of a Node Type. TOSCA’s use of types (think classes) and templates (think instances) enables reuse.

The Node Type defines the common properties of a component and also its interfaces (lifecycle operations such as “deploy,” “patch,” “scale-out,” etc.) that
can be used to operate upon the component. Each Node Template adds additional information on top of its associated Node Type that is unique to the specific instance of that type such as constraints and the number of occurrences.

Similarly, specific relationships between Node Templates are called Relationship Templates, which can establish a relationship between any two Node Templates. Relationship Templates are directional, specifying a source and a target for each relationship, and can express additional constraints, as well. In a fashion similar to Node Templates, Relationship Templates are derived from their respective Relationship Types for the purpose of facilitating reuse, and the definitions of attributes and semantics, such as containment, dependency, attachment, and so on, are defined in the Relationship Type.

A TOSCA Topology Template can contain any combination or number of Node Templates and Relationship Templates. The system is flexible. Nodes can be grouped and more complex Service Templates can be composed of simpler Service Templates, just as complex services can be aggregations of simpler services in real life.

For example, it is quite simple in TOSCA to define a cloud service consisting of an Apache Web server hosted on a specific version of Linux, and a virtual router. The Web server could be represented by a Node Template that is an instance of a Node Type of called “Web server,” with information including its name, hostname, version, and other configuration information. Similarly, the Node Template for the Linux implementation could be an instance of a Node Type called “OS,” and the virtual router a specific instance of a Node Type similarly named “virtual router,” perhaps. A Relationship Template could be created, based on a “HostedOn” Relationship Type that defines the Web Server is hosted on a specific version of Linux.
TOSCA also has a powerful mechanism to express the requirements and
capabilities of Node Templates that enables cloud provider environments that
support TOSCA to match a Node Template’s requirements, such as memory,
bandwidth, storage, versions, etc. with resources contained within the cloud
provider environment or resources further defined and provided within the cloud
service and its corresponding Topology Template itself (or even in other Service
Templates). In this way, a cloud service deployed in accordance with its service
template is guaranteed to have its requirements met (at least as far as they are
specified in the service template).

Within a Service Template, there are some types and templates such as those
related to policy that are openly defined, essentially placeholders, while others
are beyond the scope of this introduction. It can be expected that these XML
definitions, while human readable, are much more likely to be created and
manipulated by modeling tools, which will most often represent Service
Templates in visual terms. That said; the below pseudo schema from the
specification should provide a good understanding of the basic format and
construction:

```xml
01 <ServiceTemplate id="xs:ID"
02   name="xs:string"?
03   targetNamespace="xs:anyURI"
04   substitutableNodeType="xs:QName"/>
05
06 <Extensions>
07   <Extension namespace="xs:anyURI"
08     mustUnderstand="yes|no"?/>
09   </Extensions> ?
10
11 <Import namespace="xs:anyURI"?
12     location="xs:anyURI"?
13     importType="xs:anyURI"/> *
14
15 <Tags>
16   <Tag name="xs:string" value="xs:string"/>
17   </Tags> ?
18
19 <BoundaryDefinitions>
20   <Properties>
21    XML fragment
22     <PropertyMappings>
23       <PropertyMapping serviceTemplatePropertyRef="xs:string"
24         targetObjectRef="xs:IDREF"
25         targetPropertyRef="xs:IDREF"/> *
26     </PropertyMappings/> ?
27   </Properties> ?
28
29 <PropertyConstraints>
30   <PropertyConstraint property="xs:string"
31     constraintType="xs:anyURI"/>
32   </PropertyConstraint>
33 </PropertyConstraints> ?
34
35 <Requirements>
36   <Requirement name="xs:string" ref="xs:IDREF"/>
37   </Requirements> ?
38
39 <Capabilities>
40   <Capability name="xs:string" ref="xs:IDREF"/> *
41   </Capabilities> ?
42
43 <Policies>
44   <Policy name="xs:string" type="xs:anyURI">
45     policy specific content ?
46   </Policy> *
47 </Policies> ?
```
The following pseudo schema defines the XML syntax of a ServiceTemplate document (from TOSCA Specification Working Draft 12):}

Lastly, since Cloud services typically consist of virtual and physical software components such as virtual appliances, OVF files, EJBs, configuration files, ISO binary files, and so on, TOSCA provides a simple archive format, called the CSAR (Cloud Service Archive), based on the popular ZIP format, so that these various Deployment Artifacts, as they are known in TOSCA, as well as Installation Artifacts such as BPMN scripts and configuration files, and other files can be conveniently bundled and delivered to TOSCA-compliant cloud providers and tools along with the Service Template’s Topology Template.

**Conclusion**

Since TOSCA is designed to substantially facilitate the portability and lifecycle management of cloud services as hosted on a wide range of virtual and physical hardware, and on the very diverse range of complex software infrastructure available in today’s SaaS, PaaS and IaaS cloud providers, cloud service owners TOSCA is designed to substantially facilitate the portability and lifecycle management of cloud ... in today’s SaaS, PaaS and IaaS cloud providers.
and developers will be able to avoid a significant portion of the expense, delay, and overhead associated with the deployment and management of services across diverse cloud providers throughout their service lifecycle, e.g., installation, scaling in or out, patching, and removal by utilizing TOSCA-enabled modeling and optimization tools, automation, and service provider environments (containers).

It is the author’s opinion that while proprietary “manifests” and “templates” from vendors and open source communities may be developed to offer some of the same functionality, the international standardization process enables the development of a truly neutral foundation for interoperability technologies open to all parties. The Internet itself, from HTML to TCP/IP, is partly the result of such standardization. Thus, a TOSCA-based eco-system would be able to more effectively:

- Improve the freedom and efficiency of service portability between diverse cloud providers
- Facilitate cloud bursting and lower the cost of cloud service deployment and management by enabling greater efficiency, speed, and automation
- Enable easier and more reliable deployment of multi-cloud applications
- Offer a foundation for expanded, more interoperable cloud service marketplaces

None of this will happen overnight, but as real-world industry experience and further development of the TOSCA eco-system continue, models based on the TOSCA language will continue to evolve and be standardized, leading to an increasingly powerful and useful eco-system. With this continued evolution, TOSCA promises to help the cloud computing industry reach a new level of maturity and value for business, ultimately providing an important foundation for cloud computing to reach its full potential.

Cloud service owners and developers will be able to avoid a significant portion of the expense, delay, and overhead associated with the deployment and management of services across diverse cloud providers throughout their service lifecycle.

References
1. http://dmtf.org/standards/cmwg

NOTICES

Copyright © 2013 CA. All rights reserved. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies.

The information in this publication could include typographical errors or technical inaccuracies, and CA, Inc. (“CA”) and the authors assume no responsibility for its accuracy or completeness. The statements and opinions expressed in this publication are those of the authors and are not necessarily those of CA.

Certain information in this publication may outline CA’s general product direction. However, CA may make modifications to any CA product, software program, service, method or procedure described in this publication at any time without notice, and the development, release and timing of any features or functionality described in this publication remain at CA’s sole discretion. CA will support only the referenced products in accordance with (i) the documentation and specifications provided with the referenced product, and (ii) CA’s then-current maintenance and support policy for the referenced product. Notwithstanding anything in this publication to the contrary, this publication shall not: (i) constitute product documentation or specifications under any existing or future written license agreement or services agreement relating to any CA software product, or be subject to any warranty set forth in any such written agreement; (ii) serve to affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (iii) serve to amend any product documentation or specifications for any CA software product.

Any reference in this publication to third-party products and websites is provided for convenience only and shall not serve as the authors’ or CA’s endorsement of such products or websites. Your use of such products, websites, any information regarding such products or any materials provided with such products or on such websites shall be at your own risk.

To the extent permitted by applicable law, the content of this publication is provided “AS IS” without warranty of any kind, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or non-infringement. In no event will the authors or CA be liable for any loss or damage, direct or indirect, arising from or related to the use of this publication, including, without limitation, lost profits, lost investment, business interruption, goodwill or lost data, even if expressly advised in advance of the possibility of such damages. Neither the content of this publication nor any software product or service referenced herein serves as a substitute for your compliance with any laws (including but not limited to any act, statute, regulation, rule, directive, standard, policy, administrative order, executive order, and so on (collectively, “Laws”) referenced herein or otherwise or any contract obligations with any third parties. You should consult with competent legal counsel regarding any such Laws or contract obligations.