# Chapter 3
# Cloud Service Architecture and Related Standards3,2

Many enterprises plan to migrate their IT infrastructures to Cloud-based infrastructures through a phased approach. With the existing enterprise systems having arcane and inconsistent interfaces, the implementations have a tendency to develop into more complicated process flows, consisting of many subsystem interfaces to accommodate existing processes. In some cases, enterprise IT systems need to duplicate some functions to maintain consistency of business information so that enterprises can make sound financial decisions. These issues, however, are not the intent of this book. Instead, our approach is to look at the Cloud service architecture as a clean sheet scenario, peeling off issues and challenges layer by layer to reveal relevant, ultimate solutions.

While on-demand service is an outgrowth of timesharing, virtualization, and datacenters, ther Cloud service architecture is now a benchmark of new IT development. Through real or virtual agents, new generation SLAs are likely to offer a rich range of services by following mature, standardized guidance. From a user perspective, mainstream consumers will aggressively try to decrease the cost of their computing devices and be more receptive to having their client machines run free or open-source applications than the consumers currently do. Software market cycles will soon shorten due to the ease of accessibility to Cloud development platforms. Rather than the glacial pace of multi-year upgrade cycles in the current IT industry, multiple releases per year will soon become the norm. This will be rapidly accelerated even more by the development of abstracting hardware and software from the OS and software from software. All these attributes, with respect to decoupled, distributed, and mash-able "fabrics," will impact the fundamental architecture of enterprise Cloud services.

Technologically, the Cloud is a culmination of standards and technologies that have come together to form a new type of business operation. This chapter provides a view into architectural considerations and standards as they affect common architectural domains, such as enterprise, software, and infrastructure architecture. To make informed decisions and take full advantage of the potential benefits of adopting a Cloud service model, IT architects and decision makers must weigh the business drivers and technical requirements against the economic, regulatory, political, and financial landscapes surrounding the company. Industry standards that enforce the engagement of their partners and customers for business improvements will be an essential factor for their success. These include the standards of management and operation, applications, clients, platforms, services, storage, and more.

## 3.1 Overview

Based on the general considerations and visions for Cloud technology and its applications, Clouds should be uniquely identifiable so that they can be individually managed even when under federations of Clouds or when combined with other Clouds. From customers' perspective, users view the Cloud differently depending on their role within the organization. This will be necessary to distinguish and harmonize Cloud business and infrastructure policies in force. This chapter aims to systematically examine the different infrastructures and enterprise services. Figure 3.1 depicts the general infrastructure of the Cloud, which includes integration, services, and management. As the figure indicates, services are usually composed of software applications, platform services, infrastructure services, and physical infrastructure. As seen on the right side of the diagram, management aspects typically include service management (service fulfillment, service provisioning, service assurance, etc.), customer services, and information assurance.

From management's perspective, the following three characteristics are essential to any enterprise Cloud [1]:

1. Configurations are dynamic and automated (or semi-automated) in varying and unpredictable ways, and possibly even include event-driven conditions.

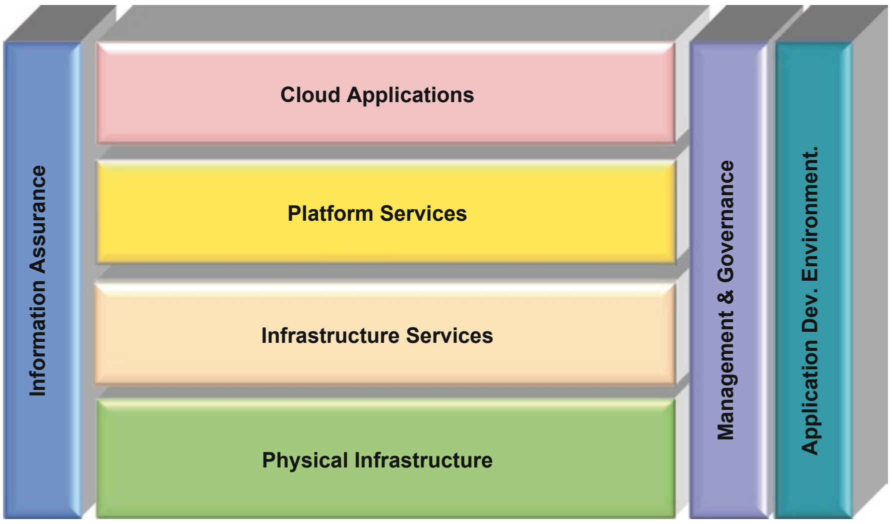

**Fig. 3.1** General Cloud infrastructure

2. Systems management technologies are scalable so that they are manageable in aggregate conditions (e.g., integration of business constraints with infrastructure constraints).
   (a) A Cloud is dynamically provisioned and able to optimize its own construction and resource consumption over time.
   (b) A Cloud is able to recover from routine and extraordinary events.
   (c) A Cloud is aware of the context in which it is used, thus the Cloud's contents dynamically behave accordingly (e.g., if Clouds are combined and composited, necessary types of policies will have to be harmonized across Cloud boundaries). Application platforms today are unaware of their usage context, however business functionality in next generation platforms will have to be managed with context in mind.
3. A Cloud is secure and has the necessary information assurance capabilities.

Cloud Computing has numerous, well-known predecessors and technologies, including utility computing, Grid Computing, virtualization, hypervisors, etc. As shown in Chap. 1, one technological concept that does not always enter the Cloud conversation, but definitely should, is SOA. SOA has played a role in enabling Cloud environments to become what they are today, and will also play a significant role in the evolution of Cloud technologies.

In many ways, Cloud Computing can be seen as an extension of SOA past applications and into application and physical infrastructure. As enterprises and Cloud providers look to provide Cloud solutions, their basic goal will be to enable the enterprise IT infrastructure as a service.

The lessons learned in integrating and providing enterprise applications as discrete services should also be applied as the infrastructure layers are organized and provided as a service. The application and physical infrastructure, much like applications in SOA, must be discoverable, manageable, and governable. Ideally, much like with SOA, open standards will evolve that dictate how the services are discovered, consumed, managed, and governed. These standards sum up the entire lifecycle of a Cloud solution [1].

Figure 3.2 captures the idea of the three-layered Cloud service approach, and shows how each of those layers are essentially offering services to an overall SOA.

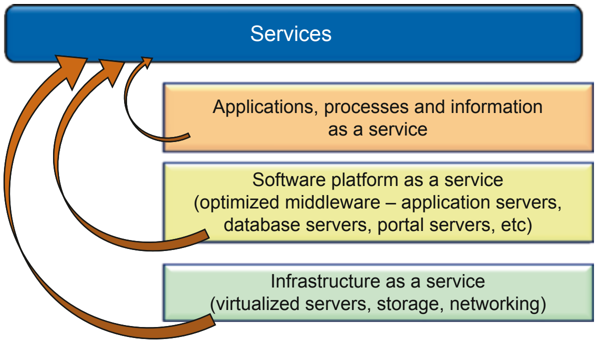
Services
Applications, processes and information as a service
Software platform as a service (optimized middleware – application servers, database servers, portal servers, etc)
Infrastructure as a service (virtualized servers, storage, networking)

**Fig. 3.2** Enterprise Cloud services

In some cases, the services in the bottom two layers are presented as part of a SOA, but the important part is the recognition of the service-based approach to all layers of the Cloud.

Cloud Computing is poised to be a significant player in the technology industry now and in the foreseeable future. In its ultimate form, it will provide the means for IT to be delivered to consumers as a service. Products and service offerings in the Cloud space continue to grow and underscore the fact that this is where things are heading. The following sections will offer a closer look at Cloud service architecture, survey some of the most related standards, and offer solutions that are moving Cloud technologies from an idea to bottom-line returns for enterprises.

As mentioned previously, commercial network architecture is typically designed with a number of horizontal network layers, each with a distinctly unique purpose. The connectivity services are typically separated from end-user services. The convergence of networks and IT, driven by Web technologies, has forced digital services into distributed computing environments. Customers are demanding SLAs at the level of distributed applications, rather than at the level of standalone products. As a result, a sophisticated mesh of revenue models directs commercial flows across the value network. The challenge to the SP is how to manage and operate the set of services and infrastructure effectively. This task involves people, processes, and systems. In the new world of distributed value chains, enterprises and providers rely upon proven and well-adopted industry standards with clearly defined procurement specifications to be agreed to with equipment and enterprises.

The majority of operational problems stem from the underlying business processes, systems, and data. In order to be competitive in this global economy, one has to react to change and bring its products to market faster and better than the competition. A holistic, service-oriented EA model is the enterprise model of choice to meet the new challenges in this evolving global economy. By leveraging and extending

Leading Organizations
DMTF
Open Cloud Consortium
cloud security alliance CSA
tmforum
SNIA
OMG OBJECT MANAGEMENT GROUP
OASIS
OGF
CLOUDCOMPUTINGINTEROPERABILITYFORUM

**Fig. 3.3** Sample industry standards and forums

industry standards and best practices, enterprises can ensure and improve interoperability, manageability, performance, scalability, and supporting service modeling and interfacing (e.g., standardized service contracts, service loose coupling, service abstraction, service reusability, service autonomy, service discoverability, service composites, etc.).

The later sections of this chapter will survey a wide range of well-known and well-documented industry standards, shown in Fig. 3.3, for a number of Cloud Computing relevant areas [2].

## 3.2 Types of Cloud Services

Cloud Computing solutions come in multiple forms: *public*, *hybrid*, *community*, *and private*. First, let us take a look at the layers of the Cloud. Figure 3.4 is a distillation of what most agree to be the three principle components of a Cloud model. This figure accurately reflects the proportions of IT mass as it relates to cost, physical space requirements, maintenance, administration, management oversight, and obsolescence. Further, these layers not only represent Cloud anatomy, they also represent IT anatomy in general.

### *3.2.1 Software as a Service*

SaaS is perhaps the most familiar to everyday Web users. The application services layer host applications that fit the SaaS model. These are applications that run in a Cloud and are provided on demand as services to users. Sometimes the services

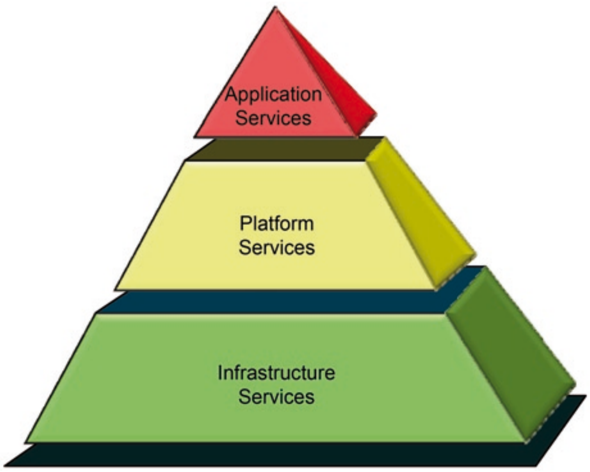

**Fig. 3.4** Types of Cloud services

are free and providers generate revenue from things like Web ads. Other times, application providers generate revenue directly from the usage of the service. This top layer of the Cloud is deeply embedded into our daily lives such as filing taxes online using Turbo Tax, check our emails using Gmail or Yahoo Mail, or keep up with appointments using Google Calendar. These are just a couple of examples of these types of applications. There are literally thousands of SaaS applications, and the number grows daily thanks to Web 2.0/3.0 technologies. Perhaps not quite as apparent to the public at large is that there are many applications in the application services layer that are directed to the enterprise community. There are hosted software offerings available that handle payroll processing, HRM, collaboration, CRM, business partner relationship management, and more. Popular examples of these offerings include IBM Lotus Live, IBM Lotus Sametime, Unyte, Salesforce.com, Sugar CRM, and WebEx. In all cases, applications delivered via the SaaS model benefit consumers by relieving them from installing and maintaining the software, and can be used through licensing models that support *pay-per-use* concepts [1, 3].

SaaS helps enterprises improve the efficiency of existing client-server applications, allowing services to be more effective over the Internet. It also expands the scope of existing web applications, whether it focuses on business-to-business or business-to-consumer applications. In order to employ SaaS, the enterprise has to first understand the complexities of delivering SaaS in a multi-customer environment. As organizations continue to adopt outsourced models for automating critical business processes, SaaS is becoming more attractive for many different types of SPs as well as ISVs. Under this model, software features can be easily enabled or disabled by customers or users based on a specific industry, work environment, or other criteria.

Through this single-source approach, SPs reduce internal operating costs and help lower the total cost of ownership for customers. Implementation time is shortened and greater user acceptance is achieved. Figure 3.5 depicts SaaS in a Cloud Computing infrastructure. Some of the challenges of implementing SaaS include [4–6] the following:

- *Multi-tenant deployment*: Multi-tenant platforms use common resources and a single instance of both the object code of an application as well as the underlying database to support multiple customers simultaneously. Current Web 2.0/3.0 deployments utilize the multi-tenant deployment, in which applications aim to facilitate collaboration and sharing between users. Very few standards have been established for multi-tenant application delivery or the operational governance to ensure isolation among customers. Questions may surface regarding the suitability of the SaaS model for mission-critical applications. Several solutions to the issues associated with multi-tenant deployment exist, namely having separate databases per customer, a shared database but separate schemas, or a shared database and shared schemas.
- *Scalability*: Given that SaaS applications are delivered via the Internet, the major challenges that apply to scalability are performance and load management. The

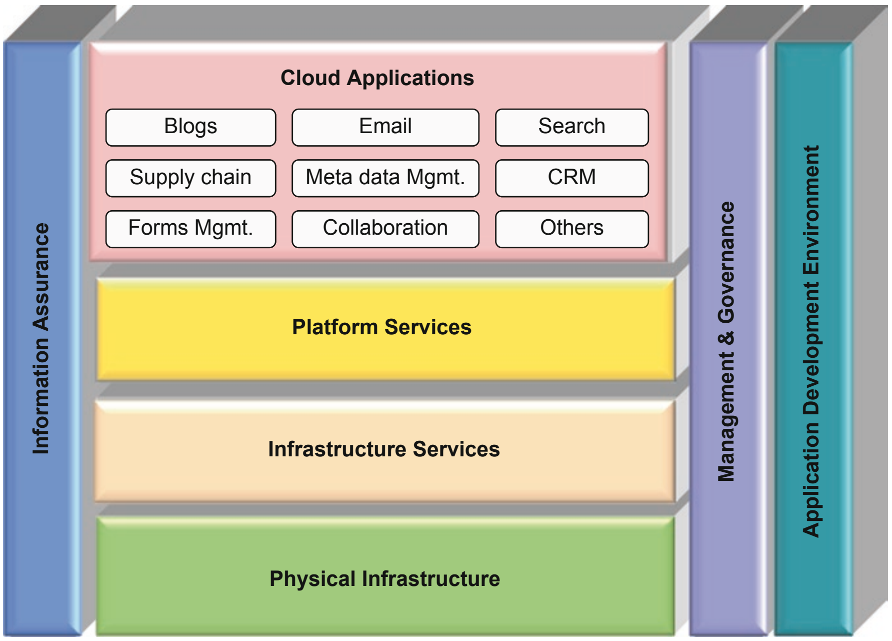

**Fig. 3.5** Cloud computing infrastructure—SaaS

designs of an application's architecture, database schema, network connectivity, available bandwidth, etc., are all effecting and complex factors of the deployment.

- *Reliability*: Reliability is the level of accuracy in which an application provides its intended services, usually dictated by user documentation or application specifications. In addition, reliability is about providing correct results and handling error detection and recovery in order to avoid failures.
- *Usability*: The trend in application development is migrating towards a more dynamic user experience. Many SaaS providers are leveraging *Asynchronous JavaScript and XML* (AJAX) to improve the overall user experience.
- *Data Security*: Data security means ensuring that data is guarded from corruption and that access to data is controlled. The very nature of SaaS poses security challenges. In order to detect and prevent intrusion, adequate strong encryption, authentication, and auditing must be a part of the application design to restrict access to private and confidential data.
- *Auditing*: Auditing involves two aspects: *auditing of security and auditing of information*. Security audits are when a third party validates a managed SP's security profile. Information audits refer to a subsystem that monitors actions to, from, and within an application.
- *Data ownership*: Data protection and ownership are probably the most difficult challenges of SaaS. The difficulty arises when the data owning party and the data safeguarding party are not the same. In addition to safeguarding data and

information, there must be a restoration procedure and corresponding disaster recovery plan in place.

- *Integration*: Integration refers to the process of combining different applications so that they work together to run smoothly as one application. As mentioned in Chap. 1, SOA is usually the approach of choice when it comes to many integration strategies.

The common application layer services provide semantic conversion between associated application processes. Examples of common application services of general interest include the virtual file, virtual terminal, and job transfer and manipulation protocols. These topics are discussed in further detail in Chap. 5.

### 3.2.2 Platform as a Service

Cloud computing has also evolved to include platforms for building and running custom applications, a concept known as PaaS. PaaS applications are also referred to as on-demand, web-based, or SaaS solutions. In the PaaS layer, application infrastructure emerges as a set of services. This includes but is not limited to *middleware as a service*, *messaging as a service*, *integration as a service*, *information as a service*, *connectivity as a service*, and so on. The services here are intended to support applications. These applications might run in the Cloud, or they might run in a more traditional enterprise datacenter. In order to achieve the scalability required within a Cloud, the different services offered here are often virtualized. Examples of offerings in this part of the Cloud include IBM WebSphere Application Server virtual images, AWS, Boomi, Cast Iron, and the Google App Engine. Platform services enable consumers to be sure that their applications are equipped to meet the needs of users by providing application infrastructure based on demand [7].

Traditionally, building and running on-premise applications has always been complex, expensive, and risky. Each application required hardware, an OS, a database, middleware, Web servers, and other software. Once the stack was assembled, a team of developers had to navigate complex programming models such as J2EE and .NET. A team of network, database, and system management experts had to be present to keep everything up and running. Inevitably, a business requirement would necessitate a change to the application, which would then kick off a lengthy development, test, and redeployment cycle. To make matters worse, large companies often need specialized facilities to house their datacenters. Enormous amounts of electricity are usually needed to power the servers as well as the systems to keep them cool. Finally, a failover site is also needed to mirror the datacenter so information can be replicated in case of a disaster. Figure 3.6 depicts PaaS in a Cloud computing infrastructure.

Just as Amazon.com, eBay, Google, Microsoft, iTunes, YouTube, etc. made it possible to access new capabilities and new markets through a web browser, PaaS offers a faster, more cost-effective model for application development and delivery.

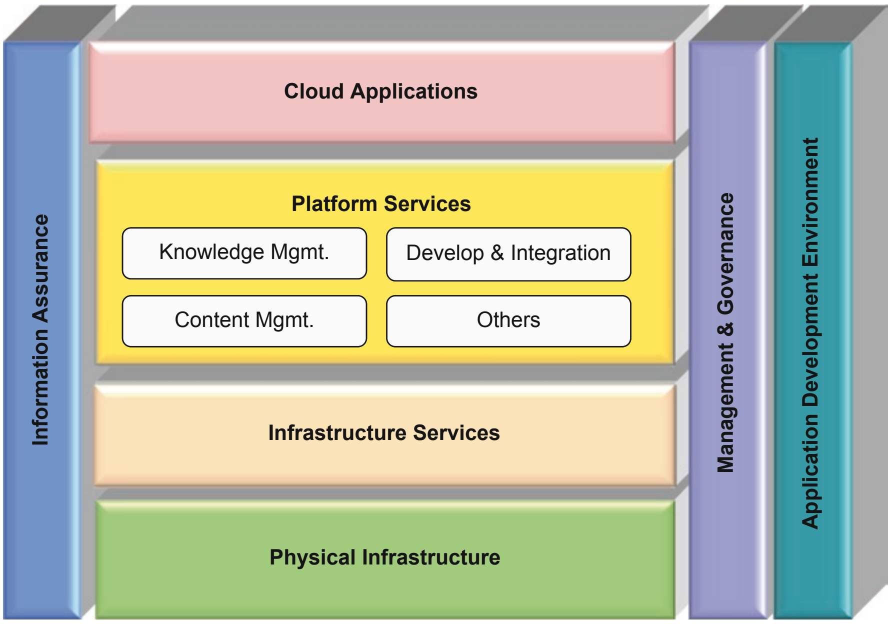

**Fig. 3.6** Cloud computing infrastructure—PaaS

PaaS provides all the infrastructure needed to run applications over the Internet. It is delivered in the same way as a utility like electricity or water. Users simply "plug in" and take what they need without worrying about the complexity behind the scenes. And like a utility, PaaS is based on a metered or subscription model, so users only pay for what they use. With PaaS, ISVs and enterprise IT departments can focus on innovation instead of complex infrastructure. By leveraging PaaS, enterprises can redirect a significant portion of their budgets from simply keeping the business running as usual to creating new and innovative applications that provide real business value. PaaS is driving a new era of mass innovation. Finally, developers can access unlimited computing power; anyone with an Internet connection can build powerful applications and easily deploy them to users wherever they are located.

An enterprise should select the platform based on its existing system landscape and skill sets, the types of applications the enterprise offers, the service delivery standards the enterprise offers, and the associated costs. Generally speaking, there are four types of platforms [8]:

- *Social application platforms*: Platforms like Facebook provide APIs so third parties can write new application functionalities that are made available to all users.
- *Web application platforms*: Platforms like Google provide APIs and functionalities for developers to build Web applications that leverage its mapping, calendar, and spreadsheets, plus YouTube and other services.

- *Business application platforms*: Platforms like Force.com provide application infrastructure specifically geared toward transactional business applications such as database, integration, workflow, and UI services. For companies unwilling to compromise on scalability, reliability, and security, Force.com is the clear choice for a flexible platform that manages critical business processes.
- *Raw computing platforms*: Platforms like AWS provide storage, processor, and bandwidth as a service. Developers can upload their traditional software stack and run their applications on the Amazon infrastructure.

According to some industry experts, more PaaS choices are available besides the do-it-yourself option, such as managed hosting, where a provider runs the infrastructure, hosts applications, and may offer SaaS-specific services. Another example is the Cloud *Integrated Development Environments (IDEs),* where applications are built using the provider's on-demand tools and collaborative development environment. In addition, many pioneer enterprises are now extending SaaS beyond single-point applications for specific needs such as sales enablement or partner relationship management. The trend is taking a broader advantage of PaaS by migrating traditional datacenter operations to less-expensive, web-centric computing environments.

### 3.2.3 Infrastructure as a Service/Hardware as a Service

IaaS or *Hardware as a Service* (HaaS) forms the bottom layer of the Cloud. A set of physical assets such as servers, network devices, and storage disks is offered as provisioned services to consumers. The services here support the application infrastructure—regardless of whether that infrastructure is being provided via a Cloud—and many more consumers. As with platform services, virtualization is often used to provide on-demand rationing of resources. Examples of infrastructure services include IBM BlueHouse, VMWare, Amazon EC2, Microsoft Azure Platform, Sun ParaScale Cloud Storage, and more. Infrastructure services address the problem of properly equipping datacenters by assuring computing power when needed. In addition, due to the fact that virtualization techniques are commonly employed in this layer, cost savings brought about by more efficient resource utilization can be realized [9, 10].

IaaS, sometimes referred to as HaaS, is another provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers, and networking components. The SP owns the equipment and is responsible for housing, running, and maintaining it. The client typically pays on a per-use basis. Characteristics and components of IaaS include the utility computing service and billing model, automation of administrative tasks, dynamic scaling, desktop virtualization, policy-based services, and Internet connectivity.

IaaS allows enterprises to scale their IT capacity up or down on command without any capital expenditure; allows data to be safely backed up and restored in

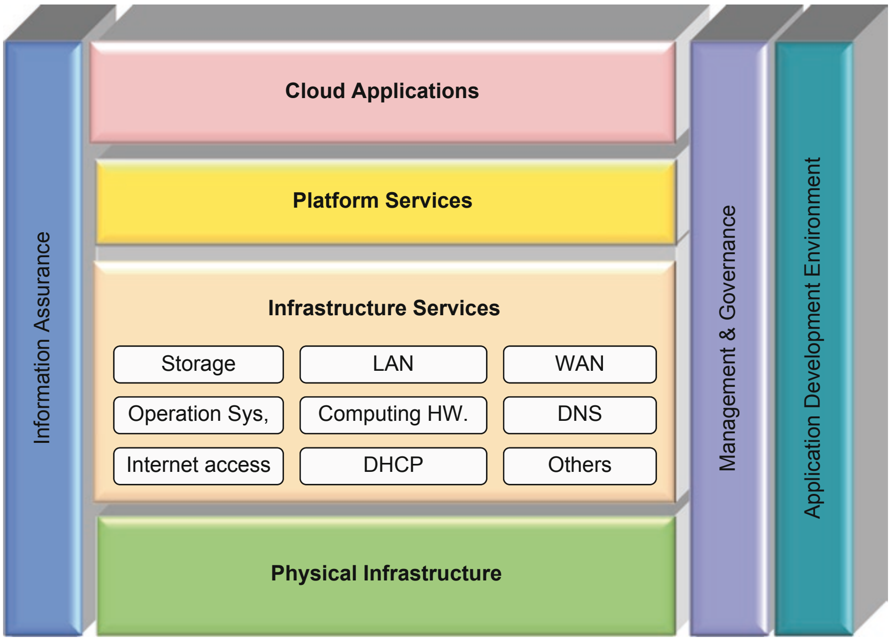

**Fig. 3.7** Cloud computing infrastructure—IaaS/HaaS

hours; and allows free, highly skilled IT staff to work on value-added tasks such as development and planning, instead of chasing bugs and installing patches ad infinitum. As a result, enterprises can significantly improve their personal market value and build a path to fulfilling a more strategic corporate role than ever before [11–13]. Figure 3.7 depicts IaaS/HaaS in a Cloud Computing infrastructure.

IaaS is enabled by a new business concept based on virtualizing the IT environment. Fundamentally, IaaS provides IT resources (processing power, storage, datacenter space, services, compliance, etc.) on-demand, enabling IT to bill these services as a variable fixed cost. The interest in IaaS can be attributed to significant increases in IT-enabled business models, such as e-commerce, Web 2.0/3.0 and SaaS, which drive demand, and by advances in technology that enable it, including virtualization, utility computing, and datacenter automation. These capabilities may enable many enterprises to better their service offerings and business efficiency. To others, it may sound like a nightmare in which they lose control of their IT environment as the computing tasks are offloaded to an outside supplier. As a result, IaaS can be viewed as a useful and enabling strategic weapon in the IT arsenal for the following reasons [14, 15]:

- *IT professionals as large-scope strategic leaders rather than micromanagers*: It is predicated by Forrester Research that "there will be more than two billion PCs in use by 2015 at a 12.3% compound annual growth rate." With that kind of explosive growth in the computer sector, it is clear that the IT administrator's scope of responsibility is going to change dramatically. The idea of a one-server-to-one-administrator model is gone. The days of logging into a single box to run patches, tweak the registry, or change permissions are gone. An ideal IT administrator/operator is someone who understands the big picture, who can grasp the importance of the 200 or 2,000 computers in use at an enterprise and how they all operate together, and can manage them as a fleet. This is precisely the IaaS model. It might sound like a significant downsizing when IT administrators can manage five to 10 times the number of devices they are managing today, however, the real equation is the availability of trained staff in today's IT market and how to best utilize their talents. What enterprises urgently need is to recruit, train, and retain IT professionals that are able to think in bigger scopes rather than micromanaging a few stations. The IT professionals of the future will need to understand how to manage hundreds or even thousands of devices. IaaS does not take away responsibility, but adds a strategic dimension to IT operations, making managers more marketable because they are now accustomed to working at a higher level.

- *IT systems must be aligned and support the business*: The purpose of IT is to conduct business more efficiently and effectively. Thus, ideally, IT would be aligned with and able to support the core business strategy of an enterprise. Today's businesses realize that IT is not just a tool to help, but is a critical part of day-to-day operations and is frequently instrumental in delivering the end product. With e-commerce, business-to-business portals, IP phone systems, and even e-mail, today's applications are fully integrated into the business, so it is critical that they behave the way the business does. With IaaS and its variable but predictable costs, it is relatively easy for any enterprise to manage spending on a monthly instead of annual basis. IaaS enables a whole new and more transparent way of accounting for IT, making precise usage and costs transparent down to the resource level (e.g., blade servers, OS, storage, etc.). This creates a closer link between what the business unit spends and the "services" it receives. Once that link is established, IT can begin to change business unit behavior to prioritize costs/benefits.
- *Choice of implementing IaaS in-house*: IaaS is both a structural concept and a mindset. Thus, it can be potentially implemented internally. It does not have to come from an outside SP. If implemented internally, the IT department can charge-back its services proportionally to the parts of the enterprise that have the heaviest users. Another new solution enabled by IaaS puts the IT budget inside other departments' budgets. In this scenario, the enterprise gives the departments dollars to spend versus IT having to carry and justify those expenses throughout the year.
- *Ability of dynamic expansion*: From a business growth standpoint, an enterprise has to be ready to expand without spending big bucks on IT resources until it is absolutely necessary. It is difficult to estimate the capital or even set it aside when the number of new customers, or whether there will be any, is unknown. Even if the money is accounted for in the budget and the enterprise does acquire new business, there is still the need to rapidly provision that environment. One of

the misconceptions about IaaS is that when an enterprise decides to use this kind of outsourcing, it is a permanent decision. IaaS is the perfect solution for an enterprise to outsource its infrastructure until the enterprise has the ability to build its own capabilities. The enterprise can have entire environments up and running in days, sometimes hours, instead of weeks. After landing a new customer, an enterprise should send it to an IaaS provider. Once the budget approval is done, the enterprise still has the choice of bringing it in-house.

- *Flexibility and scalability*: In business, opportunity implies change, and change can always be a challenge. IaaS enables rapid change because it lets companies add or remove infrastructure and services on-demand. While rapid change can impact stability, with IaaS, an enterprise can add horsepower up to 60–80% of the existing IT environment that is already stable, while gaining more control over the 20–40% that is in chaos. Imagine if an enterprise has to grow an infrastructure 10–20% in 30 days. If it decides to use a SP to help, the enterprise is not permanently stuck in that mode nor have they set a precedent for the future. One of the ideas behind IaaS is that not only can one scale up quickly, but also scale down or scale out. Furthermore, IaaS is generally delivered in addition to a utility computing platform. As long as there is a platform like VMware for virtualization, it will look identical to one's own infrastructure.
- *Datacenter automation as an integral part of IaaS*: System administrators in today's datacenters typically manage only 10–20 specific hosts or devices because they fall under the administrator's area of responsibility and/or expertise. However, as mentioned in reason 1, the administrators will soon need to manage a large number of devices and stations as a fleet, due to new technologies such as virtualization and high-density computing. Datacenter automation tools like *Opsware* (acquired by HP in 2007) and BladeLogic are a large part of the IaaS model because they enable a single administrator to manage potentially hundreds of devices. These tools provide templates and policies for configuration, patch management, and security compliance. An administrator can configure a single template based on best practices or corporate policy and apply it to several hundred machines. A delta report will show all of the devices that need attention. Built-in automated remediation lets the administrator select all of the devices and apply a single change or group of changes at once. In addition, these tools can group devices based on PBM as well as exceptions.
- *Enabling easy regulatory compliance in a federated environment*: Within a federated Cloud Computing environment, all the enterprises and SPs will have to obey the same set of regulatory rules. Using the underlying features of IaaS, compliance becomes easier. Pre-compliant VMs can be kept in a library, providing a head start when a new application environment has to be deployed. Instead of installing the server from scratch, one can deploy a copy of a pre-configured (and even pre-compliant) VM. Many organizations maintain a stockpile of prebuilt VMs in a library for this purpose. It also dramatically improves the provisioning time. After the servers are online, using datacenter automation templates, one can keep the machines in compliance and even monitor their compliance and patch-level status in a dashboard.

- *Minimization of effects and drawbacks of unexpected events*: From machine malfunctions and failures to natural disasters, there are hundreds of ways to lose data and only a few really good ways to recover it. So far, IaaS is considered the best way. Data can be backed up automatically in real-time to a strategic network of datacenters that serve as mirrored storage and backup sites. Multiple backup servers on a single physical server are possible due to virtualization, which greatly reduces the hardware and operating costs. IaaS providers generally offer these as backup "targets." Because VMs are bootable, instead of performing a bare metal restore or reinstall, all that is needed is simply to boot up the VMs, which significantly reduces the recovery time. The VMs also contain all of the precious custom configuration information that is so often lost or under-documented. At last, the use of a virtual approach also reduces the issues of hardware compatibility, as long as the VMs run on VMware and as long as VMware is installed on the recovery hardware.

## 3.3 Holistic Enterprise Architecture and Cloud Services

To achieve business modularity maturity, organizations must fundamentally shift the way they model target EA. This means a change from vertical enterprise pillars, such as process and information, product and production, IT and infrastructure, and people and organization, to a horizontal approach. Figure 3.8 shows a holistic enterprise layered perspective verses the traditional enterprise pillar perspective.

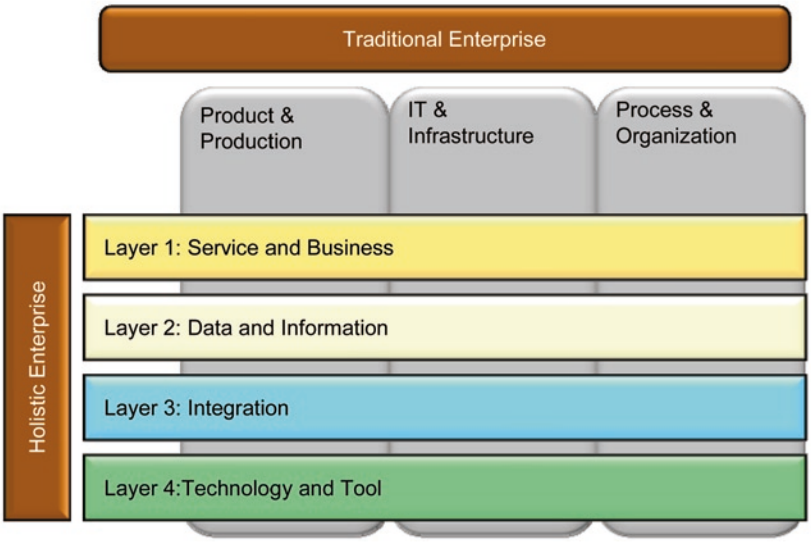

**Fig. 3.8** Holistic enterprise architecture

Chapter 5 discusses the management and technical aspects of transforming enterprises to integrate Cloud application, platform, or infrastructure services with enterprise services. In this section, we will elaborate on the holistic EA and examine the service offerings Cloud technologies provide each layer of the architecture.

### *3.3.1 Service and Business Layer*

The *Service and Business Layer* provides vision and strategic guidance, in addition to the fundamental business organization of an enterprise. The four main components of the business layer include: *business strategy*, *organization and roles*, *value network*, and the *process model*. The business layer provides the means to manage the lifecycle of business objectives and instills these objectives across the various enterprise domains. This is done by steering the establishment of the enterprise-specific process model. Furthermore, this layer also defines the roles and organizational models that take into account the extended enterprise context and integrates non-tangible concepts, such as the topology of decision making, authorization to perform pre-determined business activities, and permission to manipulate enterprise business objects. These concepts are essential to an enterprise because they govern the overall business objective and set the requirements for enterprise-wide security implementations. When integrating an enterprise with the Cloud, the Service and Business Layer often uses the Cloud SaaS. The details of this integration are in Chap. 5.

### *3.3.2 Data and Information Layer*

How data and information are stored, communicated, and interpreted, is the vital foundation of any business. They can be created, updated, and deleted only by those who are authorized to perform such manipulations within and outside of the enterprise and value network. The key function of the *Data and Information Layer* is to ensure accessibility and accuracy, and to avoid redundancy and unstructured information and data, thus optimizing effectiveness and efficiency. From a business perspective, three main components of this layer include: *semantic information definition*, *a logical data model*, *and a physical information exchange model*.

The Data and Information Layer introduces the important concept of sharing information and data among various enterprise domains. This is absolutely key in attaining the ability to accommodate a changing business environment.

In the Cloud environment, application-dependent transaction data and processes may generate data redundancy and weaken business operation efficiency and flexibility. To overcome this challenge, the Data and Information Layer must endorse the concept of enterprise-wide or cross-Cloud-wide information and data. The sharing of information and data requires two perspectives: a semantic definition of information and a data model. The semantic definition, also interpreted as the business object glossary, ensures the common understanding across different enterprise or Cloud domains. The data model perspective enables and implements information exchanges between distributed systems.

One solution to address the challenges of both the logical data model and the physical information exchange model is to create a common data model, such as the *Common Information Model* (CIM) that is developed by the DMTF and is discussed in Chap. 6. The common data model is shared and owned by a lifecycle that is managed by business processes, breaks traditional technological dependencies, and ensures the desired loose coupling of information and data with applications as well as with businesses. In addition, this common data model approach liberates data management from its traditional dependence on applications, and promotes a shared information exchange above the proprietary application data models. Another benefit of this approach is that it shifts the information and data ownership to a more business-driven lifecycle management. This makes sense, since business processes (i.e., the layer above) manage the lifecycle of business objects instead of the Technology and Tool layer. When integrating an enterprise with the Cloud, the Data and Information Layer often uses the Cloud PaaS. The details of this integration are illusatated in Chap. 5.

### *3.3.3 Integration Layer*

The two main components of the *Integration Layer* include: *process and information integration and enterprise application integration*. This layer operates the loose coupling among the logical business, the Data and Information Layer, and the Technology and Tool Layer. The main purpose is to promote the interoperability of components by conforming to standards, reducing the dependency on technologies, and ensuring scalability and responsiveness to change. It is important to note that this layer is particularly crucial to SOA rules and design principles. It is the core layer that federates different technologies and Cloud applications serving enterprise businesses.

### *3.3.4 Technology and Tool Layer*

The *Technology and Tool Layer* can be viewed as the physical layer of the holistic enterprise perspective and includes two main components: *application* and the *technological platform*. A major component in Cloud computing is tooling. In many ways, this might be the most critical to the success of a Cloud solution. There is significant technology present in the marketplace to deliver Cloud solutions, however, these technologies are often difficult to deliver due to a lack of comprehensive, understandable tooling.

Consider the application services layer in the Cloud. Tooling in this layer could provide an environment that assists with Cloud application development, and could provide the means to package and deploy the application to a Cloud infrastructure. There are already many tools that fit this description, but the problem is that they are nearly always tied to the Cloud provider's infrastructure. Open standards are key to getting the most power and flexibility from tooling. Developers cannot afford to incur the costs of learning new tools every time they switch Cloud infrastructures; further, development shops cannot continually incur the cost of rewriting applications because they switch Cloud infrastructures. For this reason, tooling should aid application development, packaging, and deployment in a way that makes the finished project portable across multiple Cloud infrastructures.

Tooling also has a very clear role in the infrastructure services layer. Building out the infrastructure for a Cloud is not a trivial process. All of the physical assets for a Cloud provider, whether that provider is internal or external, need to be considered, such that the right physical resources are allocated to the Cloud. Tools in this space should help companies visualize their IT assets so that no resources are left out of consideration for the Cloud. However, it will not be enough to provide a visualization of the assets to the Cloud constructor. The tooling in this space should offer some bit of intelligence toward the creation of the Cloud. In the past, IT administrators have had a tough job of trying to match expected demands to physical resources. This has led to the problem of under-utilization of resources. This issue is a huge catalyst for the Cloud. Tools guide users through the physical makeup of the Cloud based on the expected demand characteristics of the system. When integrating an enterprise with the Cloud, the Technology and Tool Layer often uses the Cloud IaaS/HaaS. The details of this integration are in Chap. 5.

## 3.4 Enterprise Architecture and Cloud Transformations

Cloud Computing has dramatically changed how business applications are built and run. At its core, Cloud services eliminate the costs and complexity of evaluating, buying, configuring, and managing all the hardware and software needed for enterprise applications. Instead, these applications are delivered as a service over the Internet. In this section, we will describe the architectures necessary to transform enterprises to take advantage of Cloud services [16].

### *3.4.1 Enterprise Architecture Styles*

Architecture styles define the following:

- Families of software systems in terms of patterns for characterizing how architecture components interact
- The types of architecture components that can exist in the architectures of those styles and constraints on how they may be combined
- How components may be combined together for deployment

- How units of work are managed, e.g., if they are transactional (n-phase commit)
- How functionalities that components provision may be composited into higher order functionalities, and how they can be exposed for use by human beings or other systems

There are essentially two types of architecture styles [17]: *SOA* or *non-SOA*. The *SOA* style is inherently *top-down* and emphasizes decomposition to the functional level but not lower. It is *service-oriented* rather than application-oriented, factors out policy as a first class architecture component that can be used to govern transparent performance of service-related tasks, and emphasizes the ability to adapt performance to user/business needs without having to consider the intricacies of architecture workings. Implementation of an SOA results in better architecture layering and factoring, and better interfaces that become more business than data oriented. Policy becomes more explicit and is exposed in a way that makes it easier to change it as necessary. Service orientation guides the implementation, making it more feasible to integrate and interoperate using a commodity infrastructure rather than using complex and inflexible application integration middleware.

On the other hand, the *non-SOA* (in contrast with the SOA) style is inherently *bottom-up* and takes much more of an infrastructural point of view (or *inside-out*) as a starting point, building up to a business functional layer. Application platforms constructed using *client-server*, *object-oriented*, and *tier* architecture styles are those to which the non-SOA approach is applied. This is because they form the basis of enterprise application architectures today, and because architectures of these types have limitations that require transformations to its counterpart SOA platforms.

As a rule of thumb, integrating businesses at functional levels is simpler than at lower technology layers, where implementations might vary widely. Hence, this section emphasizes decomposition to the functional level—which often is dictated by standards within a market, regulatory constraints on that market, or even accounting (e.g., Accounts Payable/Accounts Receivable/General Ledger (AP/AR/GL)) practices.

Architecture style will be critical to orchestrating services and enabling operability between thousands of collaborating businesses. Interoperability must be realized through the implementation of an architecture that integrates at a business functional level rather than a data level. Taking an SOA point of view requires a system architect or service designer to separate concerns from the start. Application platforms should be distributed from the beginning, rather than be made so after the fact by attaching some distribution layer to them. Enterprises must understand how they have permitted business security and access control models to be built into their architectures and how, now that technology innovations enable them to challenge these limits. The enterprises must remove them from their computing platforms to realize business agility goals demanded by new generation architectures. Technologies enterprises have used in the past can be useful to them in the future. Success in implementing a SOA is less a function of technology than it is of a business and technology architecture vision that forces business and technology architects to view business capabilities from a global, outside-in and top down perspective.

## *3.4.2 Architecture Transformation*

IT teams in enterprises today are under pressure to transform their existing or legacy, non-SOA, application platforms to SOA that effectively leverage the capabilities afforded through the use of Cloud and service grid technologies [17]. In this section, we will explore strategies for implementing architecture transformations from non-SOA (*bottom-up*) to SOA (*top-down*) and issues likely to be encountered in the process.

How to construct an SOA that meets modern IT computing requirements has been a topic of debate, in particular, what is the best approach to leverage past investments in infrastructure, software development, and third party software products. Furthermore, funding and how long it will take to accomplish this are other aspects being discussed. There exists a number of difficult topics that IT leaders in today's enterprises want to see addressed by Cloud and service Grid Computing, such as the following:

- Datacenter management
- Architecture transformation and evolution (evolving current architectures or beginning from scratch)
- Policy-based management of IT platforms

This section will address the architectural challenges and potential solutions. Using policy-based management mechanism to solve these challenges will be discussed in Chap. 6.

### 3.4.2.1 Transforming Existing Architectures

In the past, IT architectures took aim at the enterprise as their endpoint. Driven by new complex and dynamic business relationships, enterprises must be able to support architectures that can support entire ecosystems and, in so doing, enable these architectures to scale downward to an EA as well as upward and outward. As it has already become the critical center of business operations today, IT leaders have to continue to chase cost and margin optimization. They also have no choice but to carefully set and navigate a course to renovate and replace their existing practices and technologies with new thinking so that product lines and services that companies offer today can remain relevant through significant market transitions.

Clouds, service grids, and SOA style are technologies that will be fundamental to successful enterprise transformations. There are near term objectives, like the need for cost and resource efficiency or IT application portfolio management, that justify the use of these technologies to re-architect and modernize IT platforms and optimize the way enterprises currently deploy them. However, there are longer term business imperatives as well, like the need for a company to be agile in combining their capabilities with those of their partners by creating a distributed platform.

It is through its relevance to enterprises' existing portfolios of critical applications that real uptake will ensue, and enterprises will begin to realize the true potential of the utility model that Cloud technologies offer. Cloud technology offerings today are mostly suitable to host EAs. While these offerings provide clear benefits to enterprises by providing capabilities complementary to what they have, the fact that they can help to elastically scale EAs should not be understood to also mean that simply scaling in this way will meet modern IT computing requirements. The architecture requirements of large platforms, like social networks, are radically different from the requirements of a healthcare platform, which has geographically and corporately distributed care providers, medical devices, patients, insurance providers, etc. The requirements for these two platforms (i.e., social networks vs. healthcare platforms) are very different from those that provision straight-through processing services common in the financial services industry. Clouds will have to accommodate differences in architecture requirements like those implied here, as well as those relating to characteristics that will be discussed subsequently.

It is enticing to think that one could implement an SOA simply by wrapping an existing non-SOA application platform with Web service technologies to service-enable it. The reality may not be so simple. Technically, it is possible to wrap an non-SOA platform with Web service technologies and then evolve the non-SOA architecture to a SOA one as budget and other resources allow. Although a non-SOA might be possible to access application functionality using Web services, using the wrapper alone can not yield the benefits of a full SOA implementation. Compensation for non-SOA limits may even be more costly than taking an alternative approach.

### 3.4.2.2 Addressing Architecture Layering and Partitioning

Before laying out the plans of transforming an architecture, let us first clarify a set of architectural characteristics. These characteristics should not increase the size of the management team or other costs, should allow the system to be quickly adaptable to new technologies integrated to it, and should make the system extensible from within the enterprise out to the broader ecosystem and vice versa.

It is important that Cloud services does not realize the goals of autonomic computing as they are defined currently, though combining the characteristics of existing Clouds gets closer to this goal. This fact does not diminish their value for optimizing deployments of applications in place today. Not every Cloud needs to be autonomic, but there are benefits along each path regardless. In addition, implementing architecture features on the applications management drivers path will lead to optimizing costs of operating and maintaining automating systems management and the infrastructure and business functionalities that currently run a business, resulting in more efficient datacenter management.

Evolving an architecture toward Cloud service management and SOA capabilities can help enterprises expand their IT systems beyond enterprise boundaries.

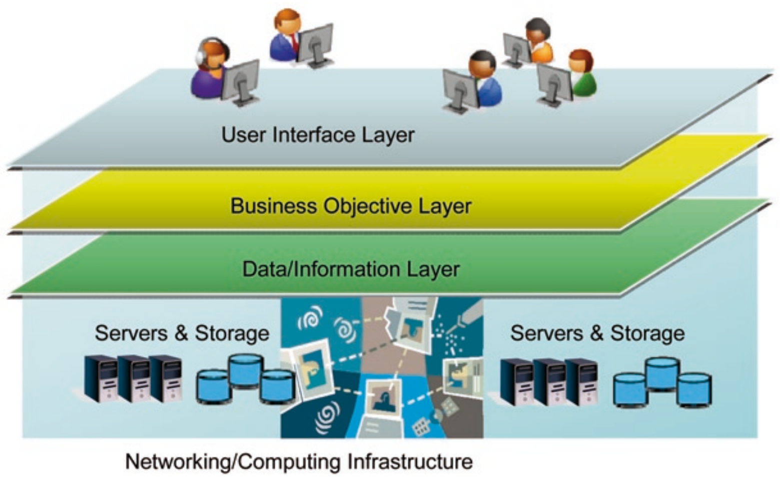

**Fig. 3.9** Web-layered architecture

This supports implementation of more flexible partner networks and value chains, but can also scale to serve virtual organizations. The first step of transitioning from one architecture style to another is to align and relate to the web application layering wherever possible. From a layered perspective, a web application is usually described with a graphic of a 3-tiered architecture, shown in Fig. 3.9, that includes a *User Interface Layer*, a *Business Objective Layer,* and a *Data/Information Layer*. The User Interface Layer is usually implemented using a web server and scripting languages. The Business Objective Layer is where all business logic programmed in various programming languages can be used to code libraries of specific, broken-down business functions. The Data Layer is where code that manipulates basic data structures goes and is usually constructed using object and/or relational database technologies. Finally, all three layers are deployed on a server configured with an OS and network infrastructure, enabling an application user to access web applications from browsers. Note that these layers correspond to the layers of the holistic EA discussed in Sect. 3.3 above.

As aforementioned, the first step in transforming an architecture is to align what an enterprise already has with the layered model, so that cross-layer violations are eliminated. An example of such an alignment could remove database specifics and business logic from the User Interface Layer. Assuming layering violations are addressed, it makes sense to introduce a service API between the User Interface Layer and the Business Objective Layer. Note that the service layer, composed of a number of services, is a means of accessing lower level functionalities. The concerns of one architecture layer do not and should not become or complicate the concerns at other levels.

Proceeding from cleaning up layering architecture violations, another important task is to clean up partitioning violations. Partitioning refers to the "componentizing" or "modularizing" of business functionalities such that a component in one business functional domain accesses functionality in another domain through a single interface. Ensuring that common interfaces are used to access business functionalities in other modules eliminates the use of private knowledge to access business functionalities in other domain spaces. Partitioning also may be referred to as factoring. Just like the previous step, the next phase of transformation focuses on partitioning functionalities in the database so that, for example, side effects of inserting data into the database in an area supporting one business domain does not also publish or otherwise impact the database supporting other business domains.

#### 3.4.2.3 Benefits of Transformations

Transforming a non-SOA to a SOA can be a lengthy process, as it is a function of existing system complexity, size, and age. Thus, it poses the question of whether or not it is worth the trouble. Clearly, it is possible to transition an architecture to become a well organized platform that is centrally hosted or hosted in a service grid or even many service grids. As a result, services and their supporting business objectives and data functionalities can be replaced easily with an alternative service implementation without negatively impacting other areas of the architecture, provided that functionality in one service domain is accessed by another service domain only through the service interface. Such a capability is required in order to simplify management of an application portfolio implemented on such an architecture, as well as distribute and federate service implementations.

When performing an architecture transformation, some of the common questions include whether or not it is necessary that all architecture components be entirely transformed; whether or not the queue-based middleware in the old architecture should be replaced; and whether or not all the old applications should be replaced with custom applications that have appropriate policy extension points. There is no fixed answer to these concerns. Certainly, it is possible to replace enterprise application integration technologies with commodity or open source technologies, simplify them, or maybe even eliminate them in some cases. It is unlikely that middleware supporting reliable messaging and long-lived business transactions between business partners needs to be totally replaced or removed from a SOA. However, its use can be couched in ways that eliminate tight coupling between partners and commingling of business policies. This is done with an integration functionality that makes partner integration difficult to change as policies change or as partner networks expand.

Cloud solutions can form the basis of an application portfolio management strategy that can be used to address tactical short term needs (e.g., interoperability within a business community of practice using the Cloud to provision community resources), and can address longer term needs to optimize the application portfolio and possibly re-architect it for the following reasons:

- Cloud vendors usually offer the capability to construct customized virtualized images that can contain software for which a enterprise has licenses. Hosting

current infrastructure in a Cloud provides an isolated area in which an enterprise and/or partners could interoperate using existing technologies.
- Cloud vendors usually offer an application functionality that can replace existing capabilities (e.g., CRM systems). Incorporating this functionality into an existing application portfolio leads to an incremental re-architecture of application integrations using newer technologies and techniques, which, in turn, should result in service-oriented interfaces that can become foundational to the future state. An incremental move toward a re-architected platform host, using Cloud technologies, may prove to be the only way to mitigate risks of architectural transformation while keeping an enterprise business running.
- Cloud APIs, together with the concepts of distribution, federation, and services that are baked in, provide a foundation on which to implement loosely coupled, SOAs and can logically lead to better architecture. Standardized interfaces, loose architecture couplings, and standardized deployment environments and methods can increase reuse potential by making it easier to compose new services with existing services.
- Clouds provide a means to deal with heterogeneity. Initially, heterogeneity is dealt with through management layers. Better architecture, as noted above, further enhances this as heterogeneity is encapsulated beneath standardized and service oriented APIs. Once heterogeneity is contained, a portfolio optimization/modernization strategy can be put into place and be implemented.

## 3.5 Cloud Architectures and Vendor Implementations

Cloud Computing instantiations are based on the following core components and technical characteristics:

- An *architecture style* (or styles) that should be used when implementing Cloud-based services
- An *external user and access control management* that enables roles and related responsibilities that serve as interface definitions that control access to and orchestrate across business functionalities
- An *interaction container* that encapsulates the infrastructure services and policy management necessary to provision interactions
- An *externalized policy management engine* that ensures that interactions conform to regulatory, business partner, and infrastructure policy constraints
- The *utility computing capabilities* necessary to manage and scale Cloud-oriented platforms.

From a Cloud solution perspective, the deployment can take one of three forms: public, community, private, and hybrid. Figure 3.10 depicts the basic concepts of these three forms. The following sections will examine the three architecture forms as they relate to an enterprise consumer of the Cloud.

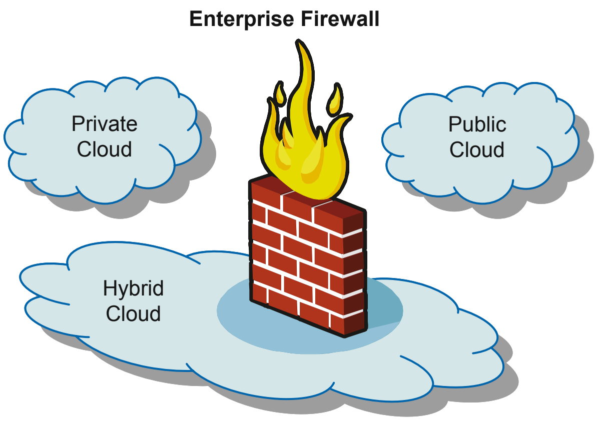

**Fig. 3.10** Three forms of Cloud computing architecture

### *3.5.1 Public Cloud*

Public Clouds are Cloud services provided by a third party (e.g., vendors or service providers). As Fig. 3.10 indicates, they exist beyond the company firewall and are fully hosted and managed by the Cloud provider. Among the three Cloud types (public, private, community, and hybrid), the Public Cloud is probably the most well-known and mature in its offerings thus far. An example of a Public Cloud is the Amazon EC2 infrastructure, which provides a Public Cloud infrastructure that hosts Amazon Machine Image instances that deliver capabilities to users [1, 18].

Accessibility and affordability are two of the key characteristics that have led to the popularity of the Public Cloud. More specifically, Public Clouds attempt to provide consumers with hassle-free IT elements. Whether it is software, application infrastructure, or physical infrastructure, the Cloud provider takes on the responsibilities of installation, management, provisioning, and maintenance. Consumers are only charged for the resources they use, so under-utilization is eliminated.

However, the Public Cloud does pose a certain degree of inconvenience, in a “convention over configuration” way. Public Cloud services are usually delivered with the idea of accommodating the most common use cases. Configuration options are usually a smaller subset than what they would be if the resources were controlled directly by consumers. Since consumers have little control over the infrastructure, processes requiring tight security and regulatory compliance require careful arrangements when using Public Clouds. These arrangements are discussed in Chap. 9.

To understand how an enterprise can leverage Public Cloud Computing solutions, let us consider two important view points. First, enterprises consume applications that are provided in the Public Cloud. This might be an application designed to process employee payroll data, or it might be a CRM system. By utilizing software delivered in this way, an enterprise can remove the burden of installing and maintaining the application on private datacenters. Another benefit is the cost savings associated with license fees, since most Cloud providers charge based on consumption. Second, enterprises utilize Cloud-based hosting solutions to deliver applications to consumers. By doing so, companies are freed from the maintenance and upkeep of production systems, since the Cloud provider is responsible for providing infrastructure resources to meet the demands users place on the application. This model also provides for an increase in the ubiquity of an enterprise's services, since solutions delivered by way of a Public Cloud can be accessed at any time from any machine with a viable network connection.

Regardless of the scenario, a common theme is the bottom line value to a business. Public Clouds can help enterprises reduce costs associated with owning software and datacenter infrastructure components. Less directly, Public Cloud usage can deliver value by enabling enterprises to respond quickly to changes in demand for their services, enabling the services to reach new markets and enabling valuable human resources to concentrate on delivering business innovation, rather than simply delivering the technological infrastructure that supports the business.

From an application provider perspective, this suite of tools lets users meet, discuss, collaborate, and innovate all by leveraging Cloud-provided services. The tools also help organizations implement solutions that leverage Public Cloud offerings in order to deliver the sought after Cloud value.

Finally, a popular Cloud services pricing structure is the *pay-per-use* structure, as discussed in Chap. 5. To achieve this, Cloud resource usage must be tracked and reported. These reports should be able to provide statistics about Cloud usage that support chargeback in the enterprise. For each user, retrieve information about their VM usage and CPU, memory, and IP utilization rates can be viewed or downloaded into a spreadsheet.

### *3.5.2 Private Cloud*

A Cloud's type is usually defined in terms of where the physical resources and data reside. Private Clouds are Cloud services provided within the enterprise. Private Clouds exist within an enterprise firewall, all of the computing resources and services that make up the Cloud are protected by that firewall [1].

Private Clouds offer many of the same benefits that Public Clouds do with one major difference: the enterprise is in charge of setting up and maintaining the Cloud. Private Cloud solutions deliver many of the same benefits as their public counterparts, such as cost reduction, business agility, and enhanced innovation. The main difference is that the enterprise maintains full control over and responsibility for the Cloud. In addition, finer-grained control over the various resources making up the Cloud gives a company all available configuration options. Private Clouds are ideal when the type of work being done is not practical for a Public Cloud, due to security and regulatory concerns.

Although a Private Cloud does not free the enterprise from the responsibility of procuring and maintaining computing resources, there are many reasons why enterprises choose Private Cloud solutions over Public Clouds:

- *Security and compliance regulations*: Private Clouds usually need more stringent control and oversight with respect to how and where data is stored than is typically provided by a Public Cloud service.
- *Capabilities that cannot be achieved in a Public Cloud*: An enterprise might require a very specific vendor technology, or might need availability guarantees not achievable by Public Cloud usage.
- *Private Cloud as financial property*: If an enterprise is heavily invested in its existing datacenter, it makes sense to optimize the utilization of those resources rather than pay for Public Cloud services. Even companies without such cost investments often see price advantages to on-premise solutions, as the flexibility of off-premise solutions could come at a premium.

One example of a Private Cloud is AWS' new service offering—the *Virtual Private Cloud* (VPC), as shown in Fig. 3.11 [19, 20]. Targeted at customers with existing IT investments, the VPC service provides a way for companies to create a logically separated set of EC2 instances and a secure VPN connection to their own networks.

Generally, VPC requires three elements: a *VPC instance*, an *IP Security* (IPSec) *VPN gateway*, and a *block of IP addresses* provided by the customer. The VPC ad-

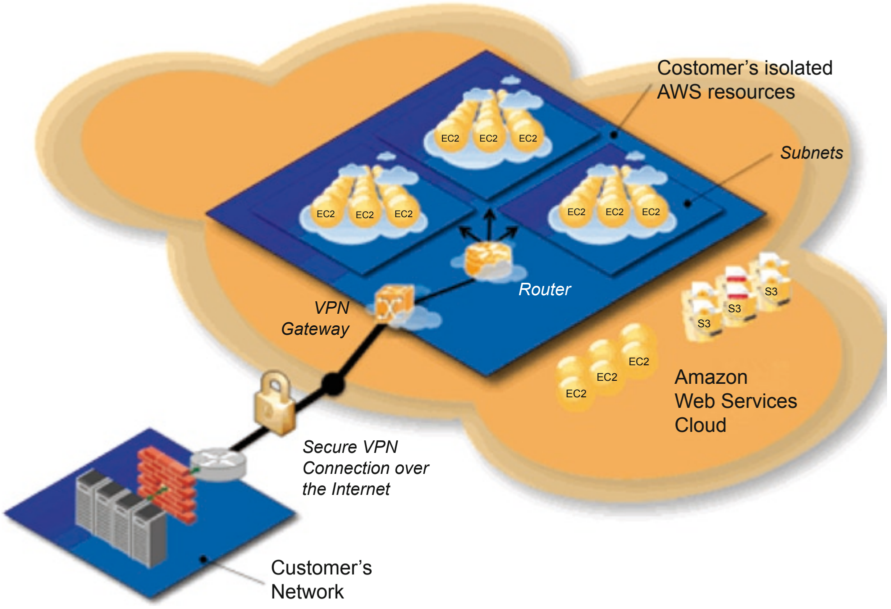
Costomer's isolated AWS resources
Subnets
EC2
EC2
EC2
EC2
EC2
EC2
EC2
EC2
EC2
Router
VPN Gateway
S3
S3
S3
EC2
EC2
EC2
Amazon Web Services Cloud
Secure VPN Connection over the Internet
Customer's Network

**Fig. 3.11** AWS virtual private Cloud

dresses can be divided up into subnets to further partition traffic. All Internet-bound traffic is routed through the customer's network and outbound security systems before reaching the public network.

Private Clouds accelerate the adoption of Cloud services if the enterprises can access a form of the Cloud that would give them the best of both worlds. This includes the flexibility and cost-effectiveness of accessing a virtually infinite pool of resources without owning it, while being able to integrate those resources into their existing datacenter environments so they could continue to leverage existing investments in their management and control infrastructure. Amazon VPC allows customers to seamlessly extend their IT infrastructure into the Cloud while maintaining the levels of isolation required for their enterprise management tools to do their work.

The Private Cloud solution potentially addresses the ever increasing costs of server and middleware management and administration in several ways. Private Clouds provide tools to build consistent, repeatable application server deployments. These deployments are optimized for virtualized environments, enabling enterprises to reduce administrative costs and leverage the benefits of server consolidation that come from such environments. In addition, the Private Cloud solution provides the flexibility to shape and tune the configurations that it dispenses. Thus, the easy integration capabilities can provide enterprises with seamless, E2E workflows that can significantly improve IT efficiency and agility even further. On the other hand, the difficulty and cost of establishing an internal Cloud can sometimes be prohibitive, and the cost of continual operation of the Cloud might exceed the cost of using a Public Cloud.

In conclusion, Private Clouds offer enterprises many of the same benefits as their public counterparts, and because of the familiarity with existing resources, Private Clouds can even provide an easier on-ramp to Cloud Computing. It provides a means to create virtualized, repeatable deployments that include everything from the OS to custom user scripts and applications. Once in the Cloud, the virtual systems can be utilized just like standard Application Server deployments.

### *3.5.3 Hybrid Cloud*

Hybrid Clouds are a combination of public and Private Clouds. These Clouds are typically created by the enterprise and management responsibilities are split between the enterprise and the Public Cloud provider. As the name suggests, a Hybrid Cloud leverages services that are in both the public and private space [1].

Hybrid Clouds are the suitable solution when an enterprise needs to employ the services of both a public and a Private Cloud. In this sense, an enterprise can outline the goals and needs of services, and obtain them from the public or Private Cloud as appropriate. A well-constructed Hybrid Cloud can service secure, mission-critical processes, such as billing and receiving customer payments, as well as those that are secondary to the business, such as employee payroll processing.

The major drawback to this type of architecture form is the difficulty in effectively creating, managing, and governing such a solution. Services from different sources must be obtained and provisioned as if they originated from a single location, and interactions between private and public components can make the implementation even more complicated. Since this is a relatively new architectural concept in Cloud Computing, best practices and tools about this pattern continue to emerge, and there could be a general reluctance to adopt this model until more is known.

Private Clouds should not be confused with Hybrid Clouds. A Hybrid Cloud uses both external (under the control of a vendor) and internal (under the control of the enterprise) capabilities to meet the needs of an application system. A Private Cloud lets the enterprise choose and control the use of,both types of resources.

## 3.6 Cloud Related Standards and Forums

In this section, a sample of standards and forums that are related to Cloud Computing and infrastructure will be discussed. As appropriate, the Cloud service (SaaS, PaaS, Iaas/HaaS) to which the standards are most applicable will be associated. Cloud Computing and infrastructure uses many more standards than what is listed in this section, and the use of these additional standards is discussed where appropriate throughout the book.

### *3.6.1 Open Grid Forum*

OGF [21] is a community-initiated forum of individual researchers and practitioners working on distributed computing, or "grid" technologies. OGF is committed to driving the rapid evolution and adoption of applied distributed computing. Applied distributed computing is critical to developing new, innovative, and scalable applications and infrastructures that are essential to productivity in the enterprise and within the science community. OGF accomplishes its work through open forums that build the community, explore trends, and share best practices, and consolidates these best practices into standards. Figure 3.12 depicts a positioning of some Cloud standards as proposed by OGF.

OGF maintains a comprehensive repository of informational, historical, and experimental documents on various topics such as Web Services Agreement Specification (WS-Agreement) [22], GLUE Schema [23], Lightweight Directory Access Protocol (LDAP) [24], Storage Resource Manager Interface (SRM) [25], Data Movement Interface (DMI) [26], GridFTP [27], OVF Specification [28], Job Submission Description Language (JSDL) [29], Basic Execution Service (BES) [30], and Usage Record (UR) [31].

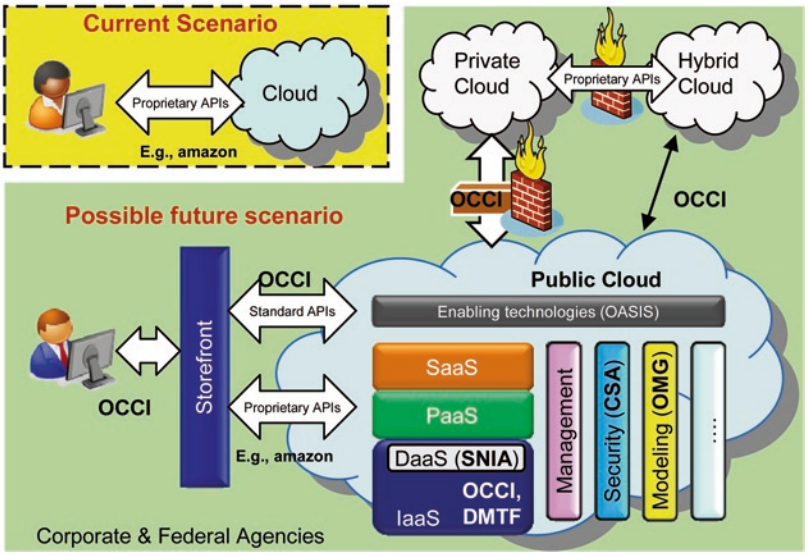

**Fig. 3.12** Possible positioning of some Cloud standards (courtesy: Dr. Craig A. Lee (OGF) brief to the CCWG on 21 Sep 2009)

### *3.6.2 Open Virtualization Format*

The OVF Specification describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of software to be run in VMs. The key properties of the format are as follows:

- *Optimized for distribution*: OVF supports content verification and integrity checking based on industry-standard *Public Key Infrastructure* (PKI), and provides a basic scheme for managing software licensing.
- *Optimized for a simple and automated user experience*: OVF supports validation of the entire package and each VM or metadata component of the OVF during the installation phases of the VM lifecycle management process. It also includes relevant, user-readable, descriptive information that a virtualization platform can use to streamline the installation experience.
- *Supports both single VM and multiple-VM configurations*: OVF supports both standard single VM packages and packages containing complex, multi-tier services consisting of multiple interdependent VMs.
- *Portable VM packaging*: OVF is virtualization platform neutral, yet also enables platform-specific enhancements to be captured. It supports the full range of virtual hard disk formats used for hypervisors today, and is extensible, which allows it to accommodate formats that may arise in the future. VM properties are captured concisely and accurately.

- *Vendor and platform independent*: OVF does not rely on the use of a specific host platform, virtualization platform, or guest OS.
- *Extensible*: OVF is immediately useful and extensible. It is designed to be extended as the industry moves forward with virtual appliance technology. It also supports and permits the encoding of vendor-specific metadata to support specific vertical markets.
- *Localizable*: OVF supports user-visible descriptions in multiple locales, and supports localization of the interactive processes during installation of an appliance. This capability allows a single packaged appliance to serve multiple market opportunities.
- *Open standard*: OVF has risen from the collaboration of key vendors in the industry, and is developed in an accepted industry forum as a future standard for portable VMs.

Virtualization and OVF are further discussed in Chap. 5. OVF is often used in the IaaS layer.

### *3.6.3 HTTP*

The HTTP [32] is a protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol that can be used for many tasks beyond its use for hypertext, such as naming servers and distributed object management systems, through an extension of its request methods, error codes, and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the WWW global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1," and is an update to the Internet Engineering Task Force (IETF) RFC 2068. The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, or protocol version, followed by a *Multipurpose Internet Mail Extensions* (MIME)-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code. This is followed by a MIME-like message from the server containing server information, entity meta-information, and possible entity-body content. HTTP is a protocol with the lightness and speed necessary for a distributed collaborative hypermedia information system. It is a generic, stateless, object-oriented protocol, which may be used for many similar tasks such as naming servers and distributing object-oriented systems, by extending the commands or "methods" used. A feature of HTTP is the negotiation of data representation, allowing systems to be built independently of the development of new advanced representations. As discussed in Chap. 5, HTTP is often used in the IaaS and SaaS layers.

### *3.6.4 XML and JSON*

XML [33] is widely used for the representation of the arbitrary data structure of Web services. XML is a text format derived from the *Standard Generalized Markup Language* (SGML). Compared to SGML, XML is simple. HTML, by comparison, is even simpler. However, XML is designed to transport and store data, not to display data like HTML.

The most recent buzz regarding XML is around its new role as an interchangeable data serialization format. XML provides two advantages as a data representation language:

- XML is text-based
- XML is position-independent

Together, these encourage a higher level of application-independence than other data-interchange formats. Unfortunately, XML is not well suited to data-interchange, much as a wrench is not well-suited to hammering nails. It carries a lot of baggage and does not match the data model of most programming languages. When most programmers saw XML for the first time, they were shocked at how ugly and inefficient it was. It turns out that that first reaction was the correct one. There is another text notation that has all of the advantages of XML, but is much better suited to data-interchange. That notation is *JavaScript Object Notation* (JSON) [34].

JSON is a lightweight data-interchange format. It is easy for humans to read and write and is also easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language-independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, structure, dictionary, hash table, keyed list, or associative array
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence

XML and JSON are used in the PaaS and SaaS layers.

### *3.6.5 AJAX*

AJAX [35] is a group of interrelated Web development techniques used on the client-side to create interactive Web applications. It is based on JavaScript and HTTP requests. AJAX is not a new programming language, but a new way to use existing standards. AJAX is the art of trading data with a Web server, and changing parts of a Web page, without reloading the whole page.

With AJAX, Web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. The use of AJAX techniques has led to an increase in interactive or dynamic interfaces on Web pages. Despite the name, the use of JavaScript and XML is not actually required, nor do the requests need to be asynchronous. In addition, AJAX.org Platform is a pure javascript application framework for creating real-time collaborative applications that run in the browser. AJAX.org Platform radically changes the way people write applications, more details can be found on AJAX.org. AJAX is used in the SaaS layer.

### *3.6.6 HTML5*

HTML5 [36] is being developed as the next major revision of HTML, the core markup language of the Web. HTML5 is the proposed next standard for HTML 4.01, XHTML 1.0 and DOM Level 2 HTML. It aims to reduce the need for proprietary plug-in-based *Rich Internet Application* (RIA) technologies such as Adobe Flash, Microsoft Silverlight, and Sun JavaFX. The ideas behind HTML5 were pioneered in 2004 by the *Web Hypertext Application Technology Working Group* (WHATWG). HTML5 incorporates Web Forms 2.0, another WHATWG specification. The HTML5 specification was adopted as the starting point of the work of the new HTML working group of the *World Wide Web Consortium* (W3C) in 2007. HTML 5 contains several features that address the challenge of building Web applications that work while offline. This document highlights these features (SQL and offline application caching APIs, as well as online/offline events, status, and the localStorage API) from HTML 5 and provides brief tutorials on how these features might be used to create Web applications that work offline.

In this 5th version of HTML, new features are introduced to help Web application authors, new elements are introduced based on research into prevailing authoring practices, and special attention has been given to defining clear conformance criteria for user agents in an effort to improve interoperability.

Users of typical online Web applications are only able to use the applications while they have a connection to the Internet. When they go offline, they can no longer check their e-mail, browse their calendar appointments, or prepare presentations with their online tools. Meanwhile, native applications provide those features: e-mail clients cache folders locally, calendars store their events locally, and presentation packages store their data files locally. In addition, while offline, users are dependent on their HTTP cache to obtain the application, since they cannot contact the server to get the latest copy. HTML5 is used in the SaaS layer.

### *3.6.7 Web Syndication*

Web syndication [37, 38] is a form of syndication in which Website material is made available to multiple sites. Most commonly, Web syndication refers to making Web feeds available from a site in order to provide other people with a summary of the Website's recently added content, such as the latest news or forum posts. The term can also be used to describe other kinds of licensing Website content so that other Websites can use it.

In addition to freely distributed material, some broadcasters and others use similar methods for the controlled placement of proprietary content on multiple partnering Internet destinations. In addition to Web feeds, commercial syndicators may use other methods to distribute their content such as Reuters, Associated Press, and All Headline News. Such commercial Web syndication borrows its business models from syndication in other media, such as Print, radio, and television. Primarily, syndication arose in those other media so that content creators could reach a wider audience. Generally, commercial Web syndication can be categorized in three ways: *business models*, *types of content*, or *methods for selecting distribution partners*.

Commercial Web syndication involves partnerships between content producers and distribution outlets. There are different structures of partnership agreements. One such structure is licensing content, in which distribution partners pay a fee to the content creators for the right to publish the content. Another structure is ad-supported content, in which publishers share revenues derived from advertising on syndicated content with that content's producer. A third structure is free or barter syndication, in which no currency changes hands between publishers and content producers. This requires the content producers to generate revenue from another source, such as embedded advertising or subscriptions. Alternatively, they could distribute content without remuneration. Typically, those who create and distribute content for free are promotional entities, vanity publishers, or government entities.

Types of content syndicated include *Really Simple Syndication* (RSS) or Atom feeds and full content. RSS is a syndication format that was developed by Netscape in 1999 and became very popular for aggregating updates to blogs and news sites. RSS also stands for "Rich Site Summary" and "RDF Site Summary." Atom is a XML-based syndication format that is used to publish headlines of the latest updates on blogs and Websites for retrieval by users and other sites. Based on RSS 2.0, Atom was turned over to the IETF for standardization. Most news aggregators support Atom along with the traditional RSS formats. With RSS feeds, headlines, and summaries, sometimes a modified version of the original content is displayed on users' feed readers. With full content, the entire content, which might be text, audio, video, applications/widgets or user-generated content, appears unaltered on the publisher's site.

There are two methods for selecting distribution partners. The content creator can hand-pick syndication partners based on specific criteria, such as the size or quality of their audiences. Alternatively, the content creator can allow publisher sites or users to "opt in" to carrying the content through an automated system. Some of these automated "content marketplace" systems involve careful screening of potential publishers by the content creator to ensure that the material does not end up in an inappropriate environment.

Just as syndication is a source of profit for TV and radio producers, it also functions to maximize profit for Internet content producers. As the Internet increases in size, it has become increasingly difficult for content producers to aggregate a sufficiently large audience to support the creation of high-quality content. Syndication enables content creators to amortize the cost of producing content by licensing it across multiple publishers or by maximizing distribution of advertising-supported content. However, a potential drawback for content creators is that they can lose control over the presentation of their content when they syndicate it to other parties.

Distribution partners benefit by receiving content either at a discounted price, or for free. One potential drawback for publishers, however, is that because the content is duplicated on other publisher sites, they cannot have the content exclusively. For users, the fact that syndication enables the production and maintenance of content allows them to find and consume content on the Internet. One potential drawback for them is that they may run into duplicate content, which could be annoying.

JavaScript is typically a useful tool for syndicating content to other Websites. Using JavaScript has the following benefits:

- *Simple implementation given the target Website*: Just add one line of HTML to the target page. It works for any Web server environment and does not need server-side technologies such as PHP, Perl, Python, or Java.
- *Real-time content update*: When updating the content on the site, changes are immediately reflected on syndication sites. With cached solutions such as RSS, there is typically a self-imposed delay of up to an hour.
- *Full control over the content*: The publisher has control over how the content is presented, or can allow partners to customize the presentation.
- *Easy viewer management*: Publishers can log information about the end-users who see their syndicated content. They can log each user that fetches their JavaScript file and then compare those results to the number of click-throughs they receive for their syndicated content.

However, using JavaScript for Web syndication also has some inherent weaknesses:

- *Dependence on the users' browsers*: If JavaScript is not enabled, the content will not appear. People with disabilities will not have access to the content, and search engines will not index the content.
- *Inefficiency*: The content must be loaded from a central location for every user. This might lead to bandwidth problems when serving the content.

As will be discussed in Chap. 5, Web syndication is appropriate for the SaaS layer.

### *3.6.8 XMPP*

The *Extensible Messaging and Presence Protocol* (XMPP) [39] is an open technology for real-time communication, which powers a wide range of applications including *Instant Messaging* (IM), presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data.

Although the core technology behind XMPP is stable, the XMPP community continues to define various XMPP extensions through an open standards process run by the XMPP Standards Foundation. There is also an active community of open-source and commercial developers, who produce a wide variety of XMPP-based software. Users are not "locked in" when using XMPP technologies. Since mid-2001, the XMPP Standards Foundation (formerly the Jabber Software Foundation) has documented and managed the Jabber/XMPP protocols through an open standards process focused on the discussion and advancement of XEPs. Such specifications define XMPP extensions and are not to be considered part of XMPP, which is only the core specifications produced by the IETF. XMPP is used in the SaaS layer.

### *3.6.9 REST*

*Representational State Transfer* (REST) is a style of architecture based on a set of principles that describe how networked resources are defined and addressed. REST is a term coined by Roy Fielding in his Ph.D. dissertation [40] to describe an architecture style of networked systems. Use of REST APIs is further discussed in Chap. 5.

The design rationale behind Web architecture can be described as an architectural style consisting of the set of constraints applied to elements within the architecture. By examining the impact of each constraint as it is added to the evolving style, the properties induced by the Web's constraints can be easily identified. Additional constraints can then be applied to form a new architectural style that better reflects the desired properties of a modern Web architecture. This section provides a general overview of REST by walking through the process of deriving it as an architectural style. Later sections will describe in more detail the specific constraints that compose the REST style.

An application or architecture considered RESTful or REST-style is characterized by:

- State and functionality are divided into distributed resources
- Every resource is uniquely addressable using a uniform and minimal set of commands (typically using HTTP commands of GET, POST, PUT, or DELETE over the Internet)
- The protocol is client/server, stateless, layered, and supports caching

The motivation for REST was to capture the characteristics of the Web that made it successful. Subsequently, these characteristics are being used to guide the evolution of the Web. REST is an architectural style, not a standard. There is not, nor will be, a REST specification. On the other hand, REST does use standards such as HTTP, URL, XML, HTML, GIF, JPEG, etc., for resource representations. REST-style Cloud configuration management is discussed in detail in Chap. 7. As discussed in Chap. 5, REST is used in the SaaS, PaaS, and IaaS layers.

### *3.6.10 Security and Data Privacy Standards*

Security and data privacy standards can be broken down into a few different categories:

- IAM
  - *Identification Management* (IdM), federation SAML, *WS-Federation*, *Liberty Identity Federation Framework* (ID-FF))
  - Strong authentication standards (*HMAC-based One Time Password* (HOTP), *OATH Challenge Response Algorithms* (OCRA), *Time-based One Time Password* (TOTP))
  - Entitlement management (*eXtensible Access Control Markup Language* (XACML))
- Data Encryption (at-rest, in-flight), Key Management
  - Public Key Infrastructure or PKI,
  - *Public Key Cryptography Standards* (PKCS),
  - *Provisioning of Symmetric Keys* (KEYPROV),
  - *Enterprise Key Management Infrastructure* (EKMI)
- RIM-International Organization for Standards ISO15489
- E-discovery (*Electronic Discovery Reference Model* (EDRM))

The detailed discussion of security and privacy issues will be deferred to Chap. 9. As an example however, in the following three sections, we examine three particular protocols in more detail: OAuth, OpenID, and *Secure Sockets Layer* (SSL)/*Transport Layer Security* (TLS).

#### 3.6.10.1 OAuth

The OAuth protocol [41] enables Websites, applications, and consumers to access protected resources from a Web service via an API, without requiring users to disclose their SP credentials to the consumers. More generally, OAuth creates a freely-implementable and generic methodology for API authentication. OAuth does not require a specific UI or interaction pattern, nor does it specify how SPs authenticate users, making the protocol ideally suited for cases where authentication credentials are unavailable to the consumer, such as with OpenID. The discussion on OpenID will be deferred to the next section. OAuth aims to unify the experience and implementation of delegated Web service authentication into a single, community-driven protocol. OAuth builds on existing protocols and best practices that have been independently implemented by various Websites. Open standards, supported by large and small providers alike, promote a consistent and trusted experience for both application developers and users of those applications.

The fundamental benefit of OAuth is that it allows users to share their private resources (photos, videos, contact list, bank accounts) stored on one site with another site without having to hand out their username and password. There are many reasons why one should not share private credentials. For example, giving an email account password to a social network site so it can look up associated friends is the same thing as going to dinner and giving the ATM card and PIN code to the waiter when it is time to pay. Any restaurant asking for a PIN code will go out of business, but when it comes to the Web, users put themselves at risk sharing the same private information. This is a good metaphor for OAuth from a user's perspective. Instead of giving the ATM card and PIN code, the card can double as a credit card with a signature authorization. Just like a username and password provide full access to the user's resources, the ATM card and PIN code provide the user with great control over bank accounts, much more than just charging goods. However, when the user replaces the PIN code with a signature, the card becomes very limited and can only be used for limited access.

Unlike OpenID, where users must do something first (i.e., get an OpenID identity they can use to sign-into sites), OAuth is completely transparent to users. In many cases, end-users will not know anything about OAuth, what it is or how it works. The user experience will be specific to the implementation of both the site requesting access and the one storing the resources, and will be adjusted to the device being used (Web browser, mobile phone, PDA, set-top box).

Users generally do not care about protocols and standards, they care about better experience with enhanced privacy and security. This is exactly what OAuth sets to achieve. With Web services on the rise, people expect their services to work together in order to accomplish something new. Instead of using a single site for all their online needs, users use one site for their photos, another for videos, another for email, and so on. No one site can do everything the best. In order to enable this kind of integration, sites need to access user resources from other sites, and these are often protected such as private family photos, work documents, bank records, etc.

#### 3.6.10.2 OpenID

OpenID [42] is the fast, easy, and secure way to sign in to Websites. OpenID is an open, decentralized standard for authenticating users. It can be used for access control, allowing users to log on to different services with the same digital identity where these services trust the authentication body. OpenID replaces the common login process that uses a login-name and password, by allowing a user to log in once and gain access to the resources of multiple software systems. The term OpenID can also refer to an ID used in the standard.

An OpenID is in the form of a unique URL, and is authenticated by the user's OpenID provider, i.e., the entity hosting their OpenID URL. The OpenID protocol does not rely on a central authority to authenticate a user's identity. Since neither the OpenID protocol nor Websites requiring identification may mandate a specific type of authentication, non-standard forms of authentication can be used, such as smart cards, biometrics, or ordinary passwords. In summary, OpenID has the following benefits:

- *Accelerate the sign-up process*: Most Websites ask for an extended, repetitive amount of information in order to use their application. OpenID accelerates that process by allowing users to sign in to Websites with a single click. Basic profile information (such as name, birth date, and location) can be stored through a user's OpenID and used to pre-populate registration forms, so the user spends more time engaging with a Website and less time filling out registration pages.
- *Easy maintenance of a single set of usernames/passwords*: Most Web users struggle to remember the multiple username and password combinations required to sign-in to each of their favorite Websites, and the password recovery process can be tedious. Alternatively, using the same password for all Websites poses a security risk. With OpenID, a user can use a single, existing account (from providers like Google, Yahoo, AOL, or blogs) to sign in to thousands of Websites without ever needing to create another username or password. OpenID is the safer and easier method to joining new sites.
- *No single control over online identity*: OpenID is a decentralized standard, meaning it is not controlled by any one Website or service provider. Users control how much personal information they choose to share with Websites that accept OpenIDs, and multiple OpenIDs can be used for different Websites or purposes. If a user's email (*Google*, *Yahoo*, *AOL*), photo stream (*Flickr*), or blog (*Blogger*, *Wordpress*, *LiveJournal*) serves as his/her primary online presence, OpenID allows the user to use that portable identity across the Web.
- *Minimize password security risks*: Many Web users deploy the same password across multiple Websites. Since traditional passwords are not centrally administered, if a security compromise occurs at any Website a user uses, a hacker could gain access to his/her password across multiple sites. With OpenID, passwords are never shared with any Websites, and if a compromise does occur, users can simply change the password for their OpenID, thus immediately preventing a hacker from gaining access to their accounts at any Websites they visit.

Finally, because the focus of most OpenID providers (such as Google, Yahoo, and AOL) is in identity management, they can be more thorough about protecting users' online identities. Most Website operators are less likely to be as dedicated to protecting users' identities as the OpenID providers, whose focus is on securely hosting user identities.

#### 3.6.10.3 SSL/TLS

One problem when administering a network is securing data that is sent between applications across an un-trusted network. TLS/SSL is typically used to authenticate servers and clients and then used to encrypt messages between the authenticated parties [43].

The TLS protocol, the SSL protocol, versions 2.0 and 3.0, and the *Private Communications Transport* (PCT) protocol are based on public key cryptography. A Security Channel (Schannel) authentication protocol suite provides these protocols.

All Schannel protocols use a client/server model. In the authentication process, a TLS/SSL client sends a message to a TLS/SSL server, and the server responds with the information that the server needs to authenticate itself. The client and server perform an additional exchange of session keys, and the authentication dialog ends. When authentication is completed, SSL-secured communication can begin between the server and the client using symmetric encryption keys that are established during the authentication process. For servers to authenticate to clients, TLS/SSL does not require server keys to be stored on domain controllers or in a database, such as the Microsoft Active Directory service. Clients confirm the validity of a server's credentials with a trusted root certification authority's certificate. Therefore, unless user authentication is required by the server, users do not need to establish accounts before they create a secure connection with a server.

In addition, SSL version 3, documented in an IETF draft, provides one of the most commonly available security mechanisms on the Internet. Developed by Netscape, SSL is used extensively by Web browsers to provide secure connections for transferring credit card numbers and other sensitive data. An SSL-protected HTTP transfer uses port 443 (instead of HTTP's normal port 80), and is identified with a special URL method. When an SSL session is established, the server begins by announcing a public key to the client. No encryption is in use initially, so both parties (and any eavesdropper) can read this key, however the client can now transmit information to the server in a way that no one else can decode. The client generates 46 bytes of random data, forms them into a single very large number, encrypts them with the server's public key, and sends the result to the server. Only the server, with its private key, can decode the information to determine the 46 original bytes. This shared secret is now used to generate a set of conventional cipher keys to encrypt the rest of the session.

## 3.7 Enterprise Transformation Implications

One of the goals of this book is to guide readers through occurring changes from infrastructure, developer, and end user perspectives that signal the demise of the full-featured server OS and the virtual server. Virtualization, and the large scale, multi-tenant operations model known as Cloud computing, enable IT professionals to rethink the packaging, delivery, and operation of software functionalities in extremely disruptive and beneficial ways [44].

There are some fundamental questions often asked: (1) how will Cloud computing affect the future of IT; (2) how will the role of IT and the roles within IT change as a result of the changing landscape of the technology it administers; and (3) what new applications and resulting markets are enabled by this fundamental shift in concept [45]?

First and foremost, software packaging will be application focused, not server focused. Traditionally, the focus of distributed system deployment has been the server, not the application. In the highly customized world of IT systems development before virtualization and the Cloud, servers were acquired, software was installed upon the servers in very specific ways, and the entire package was managed and monitored largely from the perspective of the server. For example, the focus is on what processes are running, how much CPU is being used, etc. As OS functionality begins to get wrapped into application containers, or moved onto the hardware circuitry itself, the focus shifts. The packaging begins to be defined in terms of application architecture, with monitoring happening from the perspective of software services and interfaces rather than the server itself. These packages can then be moved around within datacenters, or even among them, and the focus of management will remain on the application. That is not to say that no one will be watching the hardware; infrastructure operations will always be a key function within datacenters. However, outside of the datacenter operations team, it will matter much less.

Enterprise IT will have greater influence on solution architectures and force them to align better with what is offered from the Cloud. Whether or not the Cloud will stifle differentiation in software systems is debatable. As end users select SaaS applications to run core pieces of their business, meet integration and operations needs from the Cloud, and generally move from systems providers to SPs, the need to reduce customization will be strong. This will reduce costs and strengthen system survivability in the face of constant feature changes on the underlying application system.

In addition, the altered relationship between software and hardware due to the Cloud will result in new organizational structures within the IT department. Traditionally, when it comes to IT operations, specifically datacenter operations, administrative groups divide up along server, storage, and network lines of the client-server application architectures. However, this is a prime example of a time when applications were tightly coupled to the hardware on which they were deployed. This particular type of static deployment model requires particular expertise in customizing technologies in pursuit of meeting specific service-level goals. When software deployment is decoupled from the underlying hardware, it begins to allow for a re-evaluation of these operational roles. Currently, a lot of enterprises are already in a transition in this respect, with increasing reliance on roles like virtualization administrators and operations specialists to fulfill the changing trend [44, 46].

Furthermore, the changing landscape of software development platforms will result in new philosophies of software architecture, deployment, and operations. As mentioned earlier, applications instead of servers will become the focus, particularly agile applications ranging from web applications to data processing to core business systems, and will become more relevant in large-scale systems development. Thus, agility and project management will be the two major changes. Agility is measured in terms of the frequency and speed in which features and fixes are released from a SP's perspective. From an enterprise developer's perspective, agility is measured in how rapidly features and fixes iterate over the write-build-test cycle. Agile programming and project management methods make a ton of sense in the Cloud, as do service-oriented approaches to software and systems architecture.

Lastly, there will be a significant reduction in the demand for tactical systems administrators. More specifically, tactical system administrators in the traditional sense, i.e., who grabs a trouble ticket from the top of the queue, takes care of the request, closes the ticket, and repeats the cycle, will reduce in numbers significantly. The reason for this is automation. A lot of tasks, such as provisioning, failure detection/notification or even recovery, scaling, and some aspects of infrastructure management, are highly automatable. However, in certain situations, especially in the case of Private Clouds, tactical systems administrators are still needed. They are primarily needed to monitor the overall performance of applications running in the Cloud on both internal and external resources, as well as the performance of the Cloud providers themselves.

A common mistake made by many enterprises is to look for magic bullets that solve budget, agility, or performance problems. Several options can be considered such as (1) try to move all legacy infrastructure into a Cloud model at once; (2) put an ultimatum in place that demands that all new work be done in the Cloud, or (3) experiment with “baby Clouds,” small, noncritical projects that can prove both capability and economy, thus rationalizing a steady expansion into more critical application domains. For many enterprises, the third approach is more practical because it allows adopting enterprises to see what is available, and adopt those services at their own pace. In addition, Clouds are not defined by who runs them, but by the services they provide. For enterprises who use Cloud services for mission critical applications such as marketing and R&D support systems, they will always run their own infrastructure for some workloads and some data sets [47].

Successful enterprise transformation relies on effective Cloud service and customer management. In today’s ever-changing global economy, Cloud SPs have to respond to both the customer’s increased demands for superior customer service and to stiffer competition. Providers might have to expand their markets beyond their self-contained boundaries and broaden their business relationships. Enterprises now face very different regulatory environments and their business strategies and approaches to competition are quite distinct, nevertheless they share several common characteristics such as the following:

- Remain heavily dependent upon effective management of information and communications networks to stay competitive
- Adopt a service management approach to the way they run their businesses and their networks
- Move to an end-to-end process management approach developed from the customers’ point of view
- Aautomate customer care, service, and network management processes
- Integrate new OSS/BSS with legacy systems
- Focus on data services offerings
- Focus on total service performance, including customer satisfaction
- Integrate current technology and new technologies
- Emphasize more of a buy rather than a build approach that integrates systems from multiple suppliers

A number of commercial industry standards and best practices are able to find suitable solutions in addressing the challenges listed above. The well-accepted and well-adopted standards and best practices may give enterprises the tools they need to deliver a more productive environment and efficient management infrastructure. Generally speaking, there are four emerging categories of Cloud Computing standards:

1. *Meta-element association*: Defining "distributed and non-deterministic computing" from the Cloud and SOA perspective
2. *Governance*: Integrating service governance and Cloud governance domains
3. *SLAs*: Establishing agreements between Cloud service offering consumers and providers
4. *SOA, events, and agents*: Defining communication among and within Clouds between services enabled in these Clouds

*New Generation Operations Systems and Software* (NGOSS) is the TM Forum's next generation OSS initiative. It is envisioned as a comprehensive, integrated framework for developing, procuring, and deploying operational and business support systems and software. It encompasses a toolkit of industry-agreed frameworks, specifications, and guidelines that cover key business and technical areas; and aims to deliver measurable improvements in development and software integration environments. The elements of NGOSS fit together to provide an end-to-end framework for OSS/BSS development integration and operations. Elements of NGOSS may be used as an end-to-end framework, as part of a comprehensive methodology. In addition, the TM Forum Cloud services program is addressing some of the Cloud issues such as security, portability, and reliability from the traditional telecommunications and wider enterprise perspective.

## 3.7.1 Information Framework

While people spend a fair amount of time looking over the horizon at what the next industry-changing phenomenon will be and what will impact enterprises' businesses, it is easy to forget that it is the simple things enterprises have to get right in order to survive in the current and future marketplace. Simply put, although enterprises should be thinking about the emerging issues of Cloud Computing, challenges of streamlining processes, improving data integrity, and increasing customer experience are still the foundations of an enterprise's success.

Although information models are challenging and complex conceptual models, they serve as the bridge between business entities/domains in order to fuel an effective and efficient enterprise. An information architecture forms one of the cornerstones upon which a successful enterprise thrives.

NGOSS's enterprise-wide information framework, or *Shared Information and Data* Model (SID), provides more than a comprehensive CIM for the complete activities of an enterprise. It also provides a common language for software developers and integrators to use in describing management information, which in turn allows easier and more effective integration across OSS/BSS software applications provided by multiple vendors. More importantly, it provides the concepts and principles needed to define a shared information model, the elements or entities of the model, business-oriented models, as well as design-oriented models and sequence diagrams to provide a system view of the information and data.

### *3.7.2 Process Framework*

NGOSS's business process framework, commonly known as the *Enhanced Telecom Operations Map* (eTOM), defines several major business processes within and external to the enterprise. It is responsible for the framework and the common language of business processes. It can be used to catalog existing processes within a SP, act as a framework for defining scope of a software-based solution, or simply enable clearer lines of communication between a SP and a system integrator.

eTOM represents task-centric services that are modeled to encapsulate process logic or use case steps. It ties together the grouped logic or steps as a specific activity automated by the service logic. The purpose of the NGOSS eTOM framework is to continue to set a vision for the industry to compete successfully through the implementation of business process driven approaches to managing the enterprise. Approaches include ensuring integration among all vital enterprise support systems concerned with service delivery and support. The focus of the eTOM framework is on the business processes used by enterprises, the linkages between these processes, the identification of interfaces, and the use of customer, service, resource, supplier/partner, and other information by multiple processes. Bringing the ITIL and TM Forum standards into alignment will most definitely strengthen the practicality of the models for more generic business needs in SOEs.

### *3.7.3 Service Level Management*

Service performance encompasses both technical performance factors as well as the more subjective customer satisfaction. A SP, by offering various performance levels for a given service, has the capability to balance the level of performance offered against price and customer expectation. SLAs provide SPs with the opportunity to diversify their customers, build stronger long-term relationships and brand image, and maintain and grow their market share. However, the growing complexity of global services brings together a myriad of services, suppliers, and technologies, all with potentially different service requirements. A SLA is an element of a formal, negotiated contract, which documents the common understanding of all aspects of the service and the roles and responsibilities of both parties from service ordering to service termination. A SLA can include many aspects of a service, such as performance objectives, customer care procedures, billing arrangements, etc. Naturally, managing SLAs is also a multi-aspect task.

TM Forum has published tremendous amounts of work on SLA management and SLM. The documents include four volumes: overview, concepts and principles, applications and examples, and enterprise and applications. These four volumes are based on a common concept, with each volume concentrating on specific topics. The objective of this series is to assist the two parties (i.e., the end customer and the SP) in developing, provisioning, and managing SLAs by providing a practical view of the fundamental issues.

## 3.8 Conclusion

Constructing a lean and tight IT model requires significant improvements in enterprises' data quality and information integrity. Tremendous investments in application platforms in the past ten years have resulted in heterogeneous, best-of-breed application systems that have proven hard and costly to integrate within enterprise boundaries.

In the Cloud environment, the traditional computing platforms, i.e., physical desktops, laptops, and servers, will transform to a virtual computer and disappear behind a "networked Cloud." Computing services on the other hand, will be delivered in a highly scalable and elastic fashion. All together, the need for outlaying capital resources for computing power will be greatly reduced. Note that the Internet technologies and techniques that enable this conceptual transition will extend to the underlying hardware, storage, and applications. A new challenge of standardization in the Cloud environment is thus presented. There have been many efforts in standardizing and organizing different aspects of Cloud technologies, such as what was summarized in this Chapter. Although many efforts have been initiated, domain specifications for areas such as data interoperability, protocols, and processes for inter-Cloud collaboration and Cloud balancing, remain premature and thus require further investigation and development.

## References

1. Amrhein, D., Quint, S.: Cloud computing for the enterprise. Part 1: Capturing the Cloud, IBM DeveloperWorks. April 2009. http://www.ibm.com/developerworks/websphere/techjournal/0904_amrhein/0904_amrhein.html
2. Cloud-standards.org. April 2010. http://cloud-standards.org/wiki/index.php?title=Main_Page
3. SaaS.com. http://www.saas.com/inside.jsp?type=solutions&page=SolutionsOverview
4. Fishteyn, D.: Deploying Software as a Service (SaaS), white paper, SaaS.com. http://www.saas.com/homepage/pdf/SaaS.com_Whitepaper_PartI.pdf
5. SaaS-Sourcing—Why software as a service is dominating industries: TrendPOV. http://www.trendpov.com/node/1884

6. Oestreich, K.: Building a real-world iaaS cloud foundation. Cloud Comput. J. http://cloud-computing.sys-con.com/node/976449 (2009). 26 May 2009
7. What is PaaS? Salesforce.com. http://www.salesforce.com/paas/
8. Types of PaaS Solutions. Salesforce.com. http://www.salesforce.com/paas/paas-solutions/
9. Black, N.: What is SaaS? Understanding the concepts of Cloud computing. Feb 2009. http://blog.firmex.com/what-is-saas-concepts-cloud-computing
10. The Top 20 SaaS and Cloud Computing Buzzwords. SearchCRM.com. http://searchcrm.techtarget.com/feature/The-top-20-SaaS-and-cloud-computing-buzzwords
11. O'Day, P.: IaaS —Infrastructure as a service a threat or a weapon? Cloud Comput. J. http://soa.sys-con.com/node/439721 (2007). 11 Oct 2007
12. O'Day, P.: IaaS, A.: Web 2.0 allows user to bypass IT department, Bill St. Arnaud Blogspot. Oct 2007. http://billstarnaud.blogspot.com/2007_10_01_archive.html
13. Dornan, A.: Web 2.0 Aallows uUser to bBypass IT dDepartment, Bill St. Arnaud Blogspot., October 2007. http://billstarnaud.blogspot.com/2007_10_01_archive.html
14. Yates, S.: Worldwide PC adoption forecast, 2007 to 2015, Forrester Research. June 2007. http://www.forrester.com/rb/Research/worldwide_pc_adoption_forecast,_2007_to_2015/q/id/42496/t/2
15. Infrastructure as a service—Leveraging Cloud computing as a strategic weapon, Bluelock.com. Issue 16, June 2010. http://www.busmanagement.com/article/Infrastructure-as-a-Service–Levaraging-Cloud-Computing-as-a-Strategic-Weapon
16. Winans, T.B., Brown, J.S.: Moving information technology platforms to the Clouds, Deloitte, May 2009
17. Winans, T.B., Brown, J.S.: Cloud computing: A collection of working papers, Deloitte, May 2009
18. Amrhein, D.: Cloud computing for enterprise. Part 2: WebSphere sMash and DB2 express-C on the amazon EC2 Public Cloud, IBM DeveloperWorks. May 2009. http://www.ibm.com/developerworks/websphere/techjournal/0905_amrhein/0905_amrhein.html
19. Amazon Virtual Private Cloud, Amazon. http://aws.amazon.com/vpc/ (2010)
20. Urquhart, J.: Putting Amazon's spot pricing in perspective, cnet news. Dec 2009. http://news.cnet.com/wisdom-of-clouds/?keyword=Amazon+Web+Services
21. Open Grid Forum. http://www.ogf.org/ (2010)
22. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement), Grid Resource Allocation Agreement Protocol (GRAAP) WG. http://www.ogf.org/documents/GFD.107.pdf
23. Andreozzi, S., Burke, S., Ehm, F., Field, L., Galang, G., Konya, B., Litmaath, M., Millar, P., Navarro, J.P.: GLUE Specification v. 2.0, GLUE Working Group. http://www.ogf.org/documents/GFD.147.pdf
24. Zeilenga, K.: OpenLDAP Foundation, Lightweight Directory Access Protocol (LDAP): Technical specification road map, Network Working Group. http://tools.ietf.org/html/rfc4510
25. Sim, A., Shoshani, A., Badino, P., Barring, O., Baud, J.-P., Corso, E., De Witt, S., Donno, F., Gu, J., Haddox-Schatz, M., Hess, B., Jensen, J., Kowalski, A., Litmaath, M., Magnoni, L., Perelmutov, T., Petravick, D., Watson, C.: The storage resource manager interface specification Version 2.2, Grid Storage Resource Management, http://www.ogf.org/documents/GFD.129.pdf
26. Antonioletti, M., Drescher, M., Luniewski, A., Newhouse, S., Madduri, R.: OGSA-DMI Functional Specification 1.0, GWD-R-P.134. http://www.ogf.org/documents/GFD.134.pdf
27. Allcock, W.: GridFTP: Protocol extensions to FTP for the Grid, GWD-R. http://www.ogf.org/documents/GFD.20.pdf
28. Open Virtualization Format Specification. http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf
29. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: Job Submission Description Language (JSDL) Specification, Version 1.0, Job

Submission Description Language (JSDL) Specification. http://www.ogf.org/documents/GFD.136.pdf
30. Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S., Pulsipher, D., Smith, C., Theimer, M.: OGSA Basic Execution Service Version 1.0, GFD-R.108. http://www.ogf.org/documents/GFD.108.pdf
31. Mach, R., Lepro-Metz, R., Jackson, S.: Usage record—format recommendation, GFD-R-P.098. http://www.ogf.org/documents/GFD.98.pdf
32. World Wide Web Consortium (W3C). HTTP Specifications and Drafts. http://www.w3.org/Protocols/Specs.html (2002)
33. XML, Wikipedia. http://en.wikipedia.org/wiki/XML (2010)
34. JavaScript Object Notation. http://www.json.org/xml.html
35. http://www.ajax.org/#home
36. HTML5, Wikipedia. http://en.wikipedia.org/wiki/HTML5 (2010)
37. Web syndication, Wikipedia. http://en.wikipedia.org/wiki/Web_syndication (2010)
38. Riggott, M., Sutherland, J.: Layman's guide to Web syndication, Mercurytide company. http://www.mercurytide.co.uk/news/article/web-syndication/ (2006). 15 Dec 2006
39. Extensible Messaging and Presence Protocol (XMPP), XMPP Standards Foundation. http://xmpp.org/about/ (2004)
40. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. dissertation, Information and Computer Science, University of California, Irvine (2000)
41. OAuth Core 1.0 Revision A, OAuth. http://oauth.net/core/1.0a/ (2009). 24 Jun 2009
42. OpenID, Wikipedia. http://en.wikipedia.org/wiki/OpenID (2010)
43. What is TLS/SSL? Microsoft TechNet. http://technet.microsoft.com/en-us/library/cc784450(WS.10).aspx (2003). 28 Mar 2003
44. Urquhart, J.: Cloud computing and the big rethink: Part 5: CNET News. http://news.cnet.com/8301-19413_3-10377531-240.html (2009). 19 Oct 2009
45. Urquhart, J.: James Hamilton on Cloud economies of scale: CNET News. http://news.cnet.com/wisdom-of-clouds/?categoryId=9798870 (2010). 28 Apr 2010
46. Urquhart, J.: Does Cloud computing need LAMP? CNET News. http://news.cnet.com/wisdom-of-clouds/?categoryId=10093856 (2010). 23 May 2010
47. Business Process Framework concepts and Principles, TMF GB921, Release 9.0. TM Forum. 18 Aug 2010