ATIS Service Oriented Networks (SON) Assessment and Work Plan

January 2009
Abridged Version

ATIS is a technical planning and standards development organization that is committed to rapidly developing and promoting technical and operations standards for the communications and related information technologies industry worldwide using a pragmatic, flexible and open approach. Over 1,100 participants from more than 350 communications companies are active in ATIS' 22 industry committees, and its Incubator Solutions Program.

< http://www.atis.org/ >

The ATIS Service Oriented Networks (SON), Assessment and Work Plan is an ***ATIS Work Plan*** developed by the **Service Oriented Networks Focus Group** for the **TOPS Council**.

This document is a *work in progress* and subject to change.

Published by
Alliance for Telecommunications Industry Solutions
1200 G Street, NW, Suite 500
Washington, DC 20005

Table of Contents

EXECUTIVE SUMMARY

The Alliance for Telecommunication Industry Solutions (ATIS) Service-Oriented Networks Focus Group (SON-FG) has taken on a complex and challenging subject area on behalf of the ATIS TOPS Council. In scope it covers the following:

- From top-to-bottom of the technical architecture, including networks, Internet Protocol Multimedia Subsystem (IMS), Service Oriented Architecture (SOA), Web2.0, application, and user layers.
- All types of service provision including traditional communications based around network services, but also Information Technology (IT) services and virtualization; fixed and mobile communications; application services; third party services including those provided on the Web.
- Agile service creation undertaken by service providers; 3rd-parties and users themselves, including the creation of the associated service wrap.
- Different business and commercial models and approaches to service delivery, including the integration of carrier-grade services with those created with a Web2.0 philosophy.
- Interactions among service provider domains, third party domains and the user domain.

In order to cover this broad scope, SON-FG has accepted over 100 contributions from group members, each of which has undergone extensive analysis and discussion both within SON-FG and by members acting as catalysts for further debate within their own companies. These contributions provide the analysis of the main technologies concerned with service orientation; the main standards development organizations (SDOs) currently in play; and the business and commercial models that are used across the industry. This assessment and workplan is the culmination of that analysis and provides the main deliverable from the SON-FG activity. This work brings unique value by focusing on the cross domain aspects encompassing IMS, SOA and Web 2.0. SON-FG believes that there is no equivalent body of work available today in the industry, and that it represents a major contribution to the understanding of the way in which service orientation is applied to the business of communications.

SON-FG assessed the architectures provided by SDOs to find a single consistent framework that could serve the needs of the group to describe service-oriented networks. None was found that offered a complete yet concise articulation of the key aspects of SON and, therefore, the focus group has developed a framework which has been extensively used throughout the group's work. SON-FG member companies are at different stages of maturity in terms of the application of service orientation and the SON-FG framework serves as a useful frame of reference and lingua franca with which group members communicate.

SON-FG has concluded that it is not possible to deploy a service-oriented network entirely based on currently available standards. There are major gaps in the coverage of standards across the broad landscape of SON. Where standards are in place, they are often inconsistent, incomplete, or non-interoperable especially across the Telco, IT and web domains. Moreover, there is no single SDO responsible for standards across the piece. The most active SDOs (including TMF, Organization for the Advancement of Structured Information Standards (OASIS), OMA, International Telecommunication Union (ITU), and OMG) come from a variety

of standpoints and serve different types of stakeholders and as a result do not provide standard specifications that are compatible or that can inter-work.

There are critical gaps in the standardization of:

- Common service delivery platform
- Agile service creation
- Service Enabler (SE) descriptions
- Functional and non-functional aspects (and interfaces) for inter/cross domain exposure/ discovery, use, composability, and publishing of SEs
- Security and liability aspects of SE creation, use, and reuse
- SLAs/policies for implementation, including service quality management and quality of experience for the user
- Common Data Models and Name Space
- Service Oriented Infrastructure and IT virtualization
- Methods for a federated information architecture, common product catalogues, common metadata repository, and flexible billing and charging systems.
- Agile service wrap, including the construction of operational systems and processes from SEs
- Common policy reference information data model, and a common language (syntax and semantics).
- Service syndication
- Cross domain user profile data acquisition

One of the greatest challenges for communication service providers (CSPs) exists in the convergence of communication services with those available on the Web. The technologies, development practices, service philosophies, and business models are markedly different between these two worlds. We observe that traditional forms of communication (such as voice calls) are being displaced with message-based interactions around social communities (forums, blogs, social networks, etc.), especially for the young. As a result, some CSPs are seeing revenues fall and are witnessing a new generation entering the marketplace for whom the traditional Telco is not on their radar.

One way to meet this challenge is for CSPs to integrate Web-based services with those of communications, providing additional value along with reliability, security, and a common user experience. Currently, there are no standards in place to facilitate this and, in general, Web-based service providers (WSPs) display no desire to engage in the development of standards to this end. Another way for CSPs to meet this challenge is to present their services via open APIs for third parties to develop services. However, such APIs are not currently attractive to the main WSPs (e.g. Google, Yahoo, Facebook, etc.) who simply see CSPs as providers of the last mile of communications. Instead of convergence, we are seeing substitution.

As a result of these challenges, an opportunity is presented for ATIS to establish a leadership role for SON. SON-FG has identified over 30 work items that seek to a) fill the identified standards gaps, and b) provide a point of co-ordination for the overall standardization of

service-oriented networks, ensuring consistency between new activities and those already underway in SDOs.

To this end, the SON-FG recommends that ATIS establish a Forum with an objective to continue to build upon the conclusions made in the SON and previous Next Generation Networks (NGN) and Convergence reports to specifically address the application and service development space.

High priority areas of standardization for the Forum include:

- Service creation
- Common service enabler description including non-functional aspects
- Standardization of WS-* specifications
- Consistency of 3rd-party interfaces
- Service oriented infrastructure standardization
- Packaging of Operational Support Systems (OSS)/Business Support System (BSS) components as service enablers
- Common policy reference model

In addition, the Forum will identify the way in which Communication Service Provider (CSP) services, based upon service-oriented networks, can be made appealing to WSPs. It will establish a common dialogue between CSPs and WSPs, based upon standardization, to enable all parties to develop new products and services based upon the integration of service enablers from the worlds of communications and the Web. This will not necessarily be restricted to traditional telecom standards methods. The Forum should also consider approaches to standardization, where complimentary and appropriate, that are based upon agile development methods in which de-facto standards are quickly developed and adopted by participants and/or the use of communal resources as in the opensource model.

Note: The primary approach of the standardization activity of the Forum will be to support current SDO activity and on the co-ordination of that work across the standards landscape. For some ATIS members, participation in the SON Forum may require an adjustment to the profile of their standards activities more in favor of SOA and Web2.0, in order to properly resource the Forum's work.

1 INTRODUCTION

1.1 Arriving at the Service Oriented Network

As traditional Telcos move to build their next generation networks, offer next generation services, and orchestrate convergence of fixed and mobile assets, there exists a growing, and one might say necessary, dependence upon Internet Protocol (IP) and IP centric applications to deliver. This transformation of the Telco networks and services along with the competition presented by new opportunities based on internet technologies drives the industry to consider the abilities of its assets, and the various underlying technologies, to meet requirements of scalability, user expectations, and profitable business models.

Previous ATIS TOPS initiatives, including the NGN Focus Group and Exploratory Group on Convergence each touched upon some aspects of the issue at hand such as such as NGN service enablers and the user profile but, given the specific focus/objective of these previous initiatives, a specific and thorough investigation was not undertaken.

The Exploratory Group on Convergence (EGC) did note however, the following related concepts upon which the SON-FG was able to base its assumptions:

- Future work on the standardization of service capabilities and applications fundamentally rests upon the ability to move away from service and access specific silos to a horizontal approach.
- This also rests upon the user profile and exchange of user data across service providers and networks.
- There is a need for the standardization of service interaction. This requires the creation of clear definitions and application tools across service types and access networks, especially for common metadata.
- There is an overall industry need for a common coordination of service capability related standards developed in a number of standards development organizations and a reusable data structure (including the support of multiple applications) to enable them.
- There is also a need to coordinate the development of common mechanisms, application tools, automation and data needs (modeling, xml schema, etc.) among ATIS committees.

As customers demand flexible and personalized services and solutions to be delivered quickly, the Telco's traditional 18 month development cycle and mass subscription offerings no longer keep up with those services found for free and little cost through web providers.

While internet based services and applications are flexible, there is also a state of perpetual beta, meaning that quality of service, reliability, and redundancy are often not practiced with the robustness as traditional Telco services.

In order to remain viable, Telcos must take a hard look at the development practices (such as open source communities, exposition of APIs and availability of service enablers to third party developers) which are largely found in the web world, and make key decisions on how they will shape their future development models to maintain their five nines of reliability, and localized presence, while competing with the turnaround, flexibility, and scalability of the web.

To investigate this evolution of service creation and delivery across ATIS Technology and Operations (TOPS) Council commissioned the Service Oriented Networks Focus Group (SON-FG) to provide an assessment of:

- an enhanced service creation environment and programming model accessible to a large developer community,
- methods for blending capabilities from a variety of sources (e.g., Web 2.0/SOA/IMS); and
- the ability for ATIS member companies to selectively tap the capabilities of and provide capabilities to the larger community of external application developers.

1.2 Approach

It is the aim of the SON-FG to provide for the ATIS membership a context for standards, best practices for implementation, standardizations for the developments and delivery of services in the service oriented networks. Recognizing the expansiveness of the space in which the delivery and creation of services exists, the SON-FG refined its focus through a scoping activity which identified thirteen areas which were of common interest and importance to the industry. Included in this refined scope were:

- Development of a high level framework for SON.
- Refine terminology and definitions of critical aspects of work.
- Agree on criteria on how to determine what areas need standardization.
- Determine what aspects of service enablers need standardization and a method/criteria for determining what these aspects are.
- Determine what synergies and differences exist between the core existing frameworks of IMS/SOA/Web2.0.
- Identify work taking place on/affecting those areas important to ATIS/industry.
- Determine models for service development.
- Investigate the Service Creation Environment.
- Define the Service Delivery Platform.
- Investigate Service Orientation.
- BSS/OSS Issues.
- Determine industry needs for a common data model.
- Determine what industry needs are for Service Quality Management.

The results of the consideration of these areas are presented in this document. In order to determine the best path forward for the reuse of common service enablers the SON-FG

undertook a study of the commonalities, synergies, and deviance of the IMS, NGN, SOA, and Web 2.0 architectures, data models, and development environments. A clear picture of the interaction between these worlds is key to their coexistence.

Included in this study is the creation of terms of reference, as the group found that conversations are often complicated by their slightly different use within the contexts of IMS/Web2.0/NGN/and SOA, and common misunderstandings/ misinformation about each of these architectures.

In order to support a set of common terms of reference, a technical framework was required to provide the overall context for SON and to highlight how individual SON components fit together. A trawl across the industry identified a number of technical architectures but none provided a coherent but simplified approach.

As a result a SON Framework has also been developed to show the functionality of the Service Oriented Network. This framework highlights the recursive nature in which applications and resources are developed then used and reused in and across service provider, and third party domains, each of which may utilize a different architecture. The SON Framework is presented in more detail in Section 2.1.

In addition to the study of technical areas, three overarching themes were also considered including:

- The applicability of the technical development of the SON to the real world business models, objectives, and bottom lines of the Telco's business is also considered. For example, how are new services developed which attract users (and applications with their developers) and those services monetized? (See Section 7.)

- The question of policy implementation and enforcement of policy for services developed and delivered across domains. (See Section 6.)

- The ability to coordinate content ownership, and control in a multi ecosystem environment is also included. This included consideration of the requirements for certificate authorities and impact on existing trust models, particularly when considering the concept of applications as content and the exchange of service enabler information. (See Section 5.)

1.3 Intellectual Property

The SON-FG is considering standards and architectures defined by a number of SDOs as part of its assessment. ATIS already has established relationships with many of these SDOs, but not with all of them. One potential issue identified in this analysis was that some of these SDOs

had Intellectual Property Rights (IPR) policies that were dramatically different than the ATIS IPR policy. This raised the concern that by endorsing a standard developed by an SDO with a different IPR policy, we might create a conflict with the ATIS IPR policy. While no other concrete issues were identified, this was recognized as a theoretical possibility that should be explicitly addressed in this document. A full analysis of the IPR policies of all the SDOs identified here was briefly considered, but was ruled out as being far too large of a task, especially without any concrete proposals to assess it against. It was also noted that where ATIS has established relationships with an SDO, the IPR concerns have already been addressed. This concern only applies to a small subset of SDOs that ATIS has not worked closely with in the past. Therefore it was proposed that the SON FG would adopt the following position concerning IPR:

- The SON-FG will **not** consider IPR issues in its initial analysis. It will focus on a purely technical assessment of the architectures and standards, and recommend a path forward based on that assessment.
- Once the SON-FG report is complete, ATIS legal counsel will be asked to complete an assessment of the IPR impact of our proposed strategy. In particular, the implications of adopting standards defined by other SDOs should be considered. The analysis should identify any cases where the recommended technical strategy would effectively modify the ATIS IPR policy, and these will be highlighted to the TOPS Council for a decision as appropriate.
- If the SON-FG technical recommendations are rejected because of IPR issues, it will revisit the recommendations in light of the issues identified.

2 SERVICE ORIENTED NETWORKS FRAMEWORK

2.1 SON Framework

The SON Focus Group brings together a multi-facetted set of business and technical considerations when analyzing how service providers can implement the concepts of service oriented architectures, web2.0, IMS, etc into their products and services. These call upon many of the technical disciplines that exist in a service provider's portfolio, and include all of the technology areas described in Sections 3-5. In telecommunications service providers, these disciplines typically reside in different organizational units, with different expertise, that may, for example, be concerned with networks/IT or concerned with development/operations. As a result it is necessary to introduce a basic framework to articulate an integrated set of principles, enabling the SON-FG to be able to communicate with these disparate groups in a consistent way.

The framework therefore, sets out to be a 'lingua franca' upon which the rest of this paper is based. The ideas and concepts in this section are mostly well-known by the industry and are described in various forms in other frameworks, architectures, strategies and specifications from other SDOs. However, these various sources generally represent a single industry segment or technology perspective, and we have not found a single authoritative source of a concisely

described framework that serves our purpose. The framework we have developed is used throughout this document to illustrate the various pieces of analysis that the focus group has undertaken. Use Cases, showing instantiations of the framework in different business scenarios are typical examples of this.

Figure 2-1 shows a pictorial representation of the SON framework. It is partitioned into horizontal layers that represent a breakdown of the functional elements of a service-oriented network and a set of vertical domains that represent different business interests that might be involved in the supply chain for a SON product or service.

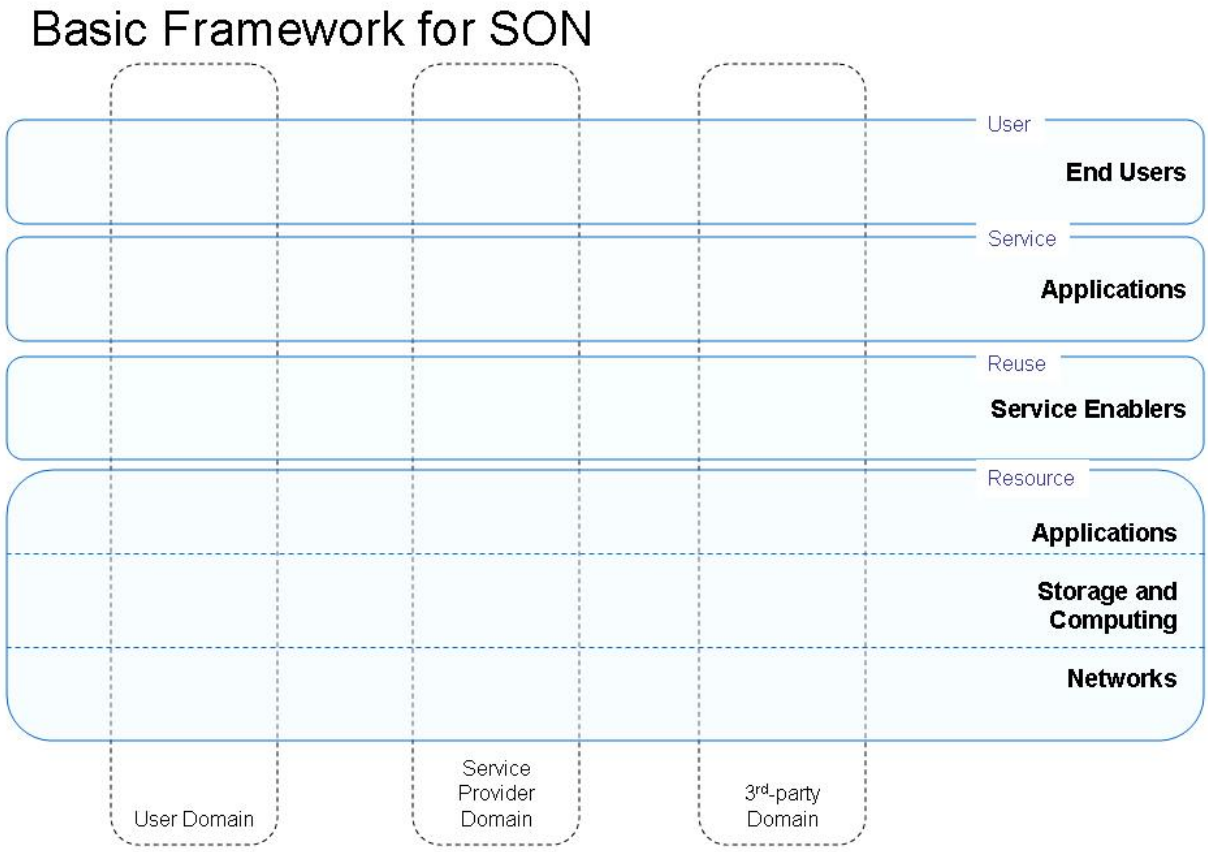


Figure 2-1: SON Framework

The lower Resource layer represents the underlying hardware and software platforms that service providers (and users themselves) deploy. This is split into networks (representing all aspects of core, backhaul and access), storage and computing (representing all aspects of IT infrastructure), and applications. This latter category represents integrated sets of software executables that are recognized as providing a bounded and distinct functionality. For example, a network resource might be a class V switch, a storage resource might be a file database, and an application resource might be a messaging application. In a service–oriented architecture all of these resources can potentially provide 'services' to other components in that architecture – one of the fundamental principles of a service-oriented network.

In order for these resources to provide such a service they must be developed as service enablers – the purpose of the next layer up. From a product perspective, these may be considered product components, but from a technical perspective, they involve the creation of discrete bundles of functionality that are exposed using a programmatic interface so that the functionality can be re-used. This element of re-use is another fundamental principle of SONs. The application of this principle is intended to allow the world of telecommunications to effectively mimic the production processes of the auto industry in which component parts of many vehicles are shared across different models. The most efficient auto manufacturers are the ones where the re-use is maximized and where the processes of integration (assembly) are agile. For SON service enablers, their functionality is 'exposed' via the programmatic interface to allow them to be used in the integration process.

The result of an assembled set of auto components is the finished vehicle itself. In a SON, the result of an integrated set of service enablers is a finished application as represented by the next layer up. This application layer consists of those software elements that a) undertake the integration (including all aspects of service logic and composition of enablers), and b) deliver the final service to the user.

Note: readers will notice that there is an application layer and an application sub-layer as part of the resource layer. This represents the fact that SONs are recursive in nature. For example, a service provider may install e-mail applications, Instant Messaging (IM) applications and voice-mail systems as resources in its data centers. It may then expose each of these as service enablers and integrate them at the application layer into a combined 'messaging' application. In a different context, this messaging application can be considered to be a resource which the SP exposes as a service enabler and integrates with a calling service enabler to create a higher value application that provides an integrated messaging and communication service.

An important consideration in the delivery of SON services is that service enablers in one domain can be used as components part in the construction of an application in a different domain. Each service provider may choose to integrate only their own service enablers or may choose to mix their own with others from third parties. There are many aspects of deploying service enablers that need to be in place whether they are to be offered internally or externally, such as lifecycle management, capacity management, reliability, etc. However, there are some key differences between exposing a service enabler internally and exposing it to 3rd-parties, including capabilities such as the ability to support the commercialization of that service enabler, and an enhanced security model. Many of the things that a service provider would do to 'productize' an internal application would also be needed to 'productize' a service enabler. Some service providers provide a Software Development Kit (SDK) to do this. The SDK allows 3rd-party developers to construct applications using their own integrated development environment (IDE) by combining the functions of their own software with those exposed by the service provider's service enablers. The SDK will also include the necessary commercial facilities to allow for the monetization of the usage of those service enablers across domains. For that purpose, a service enabler will not only expose a functional interface to the application

layer but also an interfaces for dealing with lifecycle management or charging policies, as examples.

The user layer represents the different functions that enable the service to be consumed by the user. Many of these components exist in the User domain (vertical layer) such as a home network or a user device, but others will exist in the service provider domain (such as service provider software that resides on the home network or user device).

The horizontal layers, therefore, are used to show how the functional elements of an end-to-end product or service can be divided into different categories. This split between resource, service enabler, application and user is a characteristic representation of service-oriented approaches that are well established in the industry. It provides us with a useful breakdown of a SON to allow us to assess where standardization exists or needs to exist in order for service providers to make SON a reality.

However, to complete this story, one must also consider the commercial / business and organizational boundaries that exist a) between service provider and user (although strictly the contract will be between SP and a customer), and b) between the service provider and other parties that may deliver some of the component parts. This is the purpose of the vertical domains shown in the diagram. For a complex product there may a number of parties involved in the value chain. For Internet Protocol Television (IPTV), for example, there may be a content provider, a content aggregator, an advertising broker, a transport network provider, and a broadband access provider, all providing different elements of service which the IPTV service provider integrates and delivers to its customers. These will all be represented by different instantiations of a 3rd-party domain. Any instantiation of a service provider or 3rd-party domain can contain resource, service enabler, or application components. In addition, there are many examples of where the user domain itself contains instances of these that contribute to the final service.

The combination of business domains and functional layers illustrates how the creation of applications for SON is not only an internal process to one organization, or limited to some of its resources, but is an open, collaborative process in which many entities may take part. The service creation process will encompass the capabilities of all layers, from resources to users, and will deal with functions and operations distributed across domains.

One important aspect of the delivery of products and services to customers that is not explicitly shown on the basic framework diagram is that of management (i.e. how the service wrap, OA&M, and OSS are represented). Within the overall service delivery there will be management components in all domains – service provider, 3rd-party and user. That is, management is a 'sub-domain' that will be highlighted in the use cases where the management aspects are displayed.

A further principle of SON is that the construction of a management application (e.g. an order management application that captures, validates and fulfils a customer order) follows a similar pattern to the construction of a service application (e.g. an IPTV service), in that it can be built from resources, exposed as service enablers and integrated in the application layer. Moreover, these service enablers may exist in different domains – that is, be provided by a range of service providers and/or third parties. It is the desire of telecommunications providers to produce a common set of management service enablers that can be integrated in this way. In the example of the order management application, a service provider may construct management service enablers that enact each of the major parts of the order process; that is, one for order capture, one for validation, one for fulfillment, etc. The service provider may also expose these management service enablers to others to allow them to construct their own management applications.

2.2 Use Cases

The use cases in this section have broad applicability across SON, and are complementary to the additional use cases introduced in the latter portions of this document showing more particular concepts. These use cases show the framework in practice, where the various aspects of service construction and delivery (including management) are shown divided across functional layers and the business/commercial domains.

2.2.1 Use Case – Service Enabler Creation and Deployment

2.2.1.1 **Overview**

The purpose of this use case is to illustrate the creation of a SE that could be used by a service application (SA) in its domain or another domain, and raises aspects that we may want to consider standardizing some aspect of, namely:

- how inter-domain (as well as intra-) publishing of SEs is handled.
- what kind of data associated with the SE might be included, such as what other domains may use it or conditions of use by another domain.

Here is the tangible scenario:

A SE_{α} say a location enabler, is to be created and deployed within the SP Green domain, and is also to be exposed within that domain and to the SP Blue domain, but not other domains. A service application SA_{β}, say a presence application, in the SP Blue domain will be created that will make use of SE_{α}.

2.2.1.2 **Detailed steps**

Detailed Steps:

1. The SE_α is created, say, by a SP_G Service Delivery Platform.
2. SE_α is deployed in SP_G.
3. In parallel, SE_α is published in the SP_G SE Registry with rules about to which other domains it can be exposed.
4. The existence of SE_α is made known to the SP_B Application Creation Platform.
5. The SP_B Application Creation Platform creates a service application SA_β that will use SE_α.
6. SA_β is deployed.

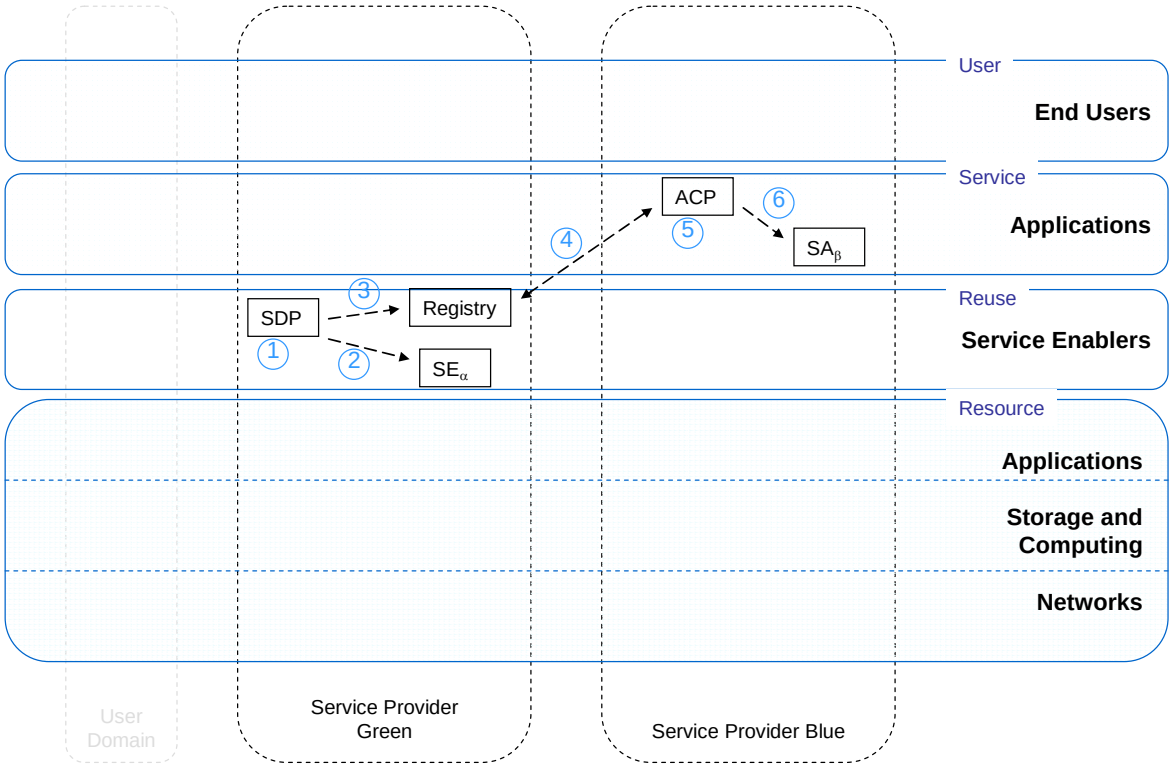


Figure 2-2: Service Enabler Creation and Deployment

2.2.1.3 Analysis in context of SON framework and standardization

The following areas may be desirable to consider for standardization:

a. How a SE is made known to an Application Creation platform; both intra- and inter-domain.
b. What the 'description' data of an SE includes and its format, beside what it does and its interfaces. E.g., conditions of use such as charging, or a description of what it does to be used in SA creation (as opposed to a description of its consumer interface).
c. How SE publishing can occur selectively to some domains, but not just any domain

d. When it comes time for an SA to use SE, what information is passed that tells the SE that that SA is a valid user?

Note that we are *not* addressing standardizing characteristics of a Service (Enabler) Delivery Platform; i.e., what its user interfaces look like, or its own operations interfaces, or how it is structured.

2.2.2 Use Case – Service Syndication

2.2.2.1 **Overview**

The basic idea is that a Service Application Creator Red (SAC_R) creates a service application and it is on-boarded into a public services marketplace. (E.g., a service application supporting residential type calling and features or a location finder for the nearest crop circles.) Service Provider Yellow (SP_Y) sees the service application and wants to purchase its use and have it configured (typically including branding) as a Yellow end-user service (e.g., Yellow Supreme Residential Service). Configuration involves selecting behavioral options and rating options and uploading some terms and conditions and marketing collateral (HTML). SP_Y will elect to use OSS/BSS Provider Orange (OP_O) to configure and manage the end-user service and to bill end-users for it.

The use case is depicted as one-stop shopping for Service Provider Yellow. I.e., the service application will run in Red's domain, and the coordination to involve Orange for management and billing is also done through Red. When end-users sign up for Yellow's service and administer their own parameters, they do it through Red interfaces that are branded Yellow.

2.2.2.2 **Detailed steps**

Steps:

1. SAC_R creates Service Application X (SA_X) and places it in SAC_R's 'Raw' Service Catalog, places it in Service Inventory.

   'Raw' means that it is not configured for use by anybody, and all its flexibility is described.

2. SAX is also deployed so that is fundamentally ready to be used, even though it still has to be configured by use for any particular service provider that would purchase its use.

3. Through its own 'Marketplace Agent', SP_Y views Red's Raw Service Catalog, and wants to use SA_X. Yellow also sees that there is an option to use OP_O for management of SA_X and for billing end-users.

4. Yellow 'purchases' SA_X and configures its use through the Marketplace Configuration Agent of SAC_R.

   It is configured so that when Yellow's end user access it, will be using a Yellow version of SA_X; i.e., configuration involves defining SP_Y-specific data for use of SA_X. Overall configuration also includes the election and specifics of the use of OP_O for its management and billing Yellow customers.

5. The Red Marketplace Config & Staging SE provides Yellow data to a Customer Profile SE.

   Note: The M C & S SE is also ultimately responsible for determining when SA_X is ready for use by Yellow, including having the management/billing support of Orange in place.

6. The Red Customer Profile SE interacts with the Orange OSS Agent for configuration information associated with use by Yellow.

7. The Orange OSS Agent makes use of the Customer Profile SE to access Yellow data relevant to its role.

8. The Customer Profile SE also makes SA_X aware of data associated with handling Yellow calls/traffic/end users/howeveryouwanttosayit. (Note: the Customer Profile SE may also have to touch other network resources such as border elements or call control elements.)

9. The Red Marketplace Config & Staging SE enters Yellow's version of the end-user service in the End-User Svc Catalog that can be viewed by potential end users.

Not described here are subsequent big pieces – end-users signing up for Yellow's service, and end-users actually using the service.

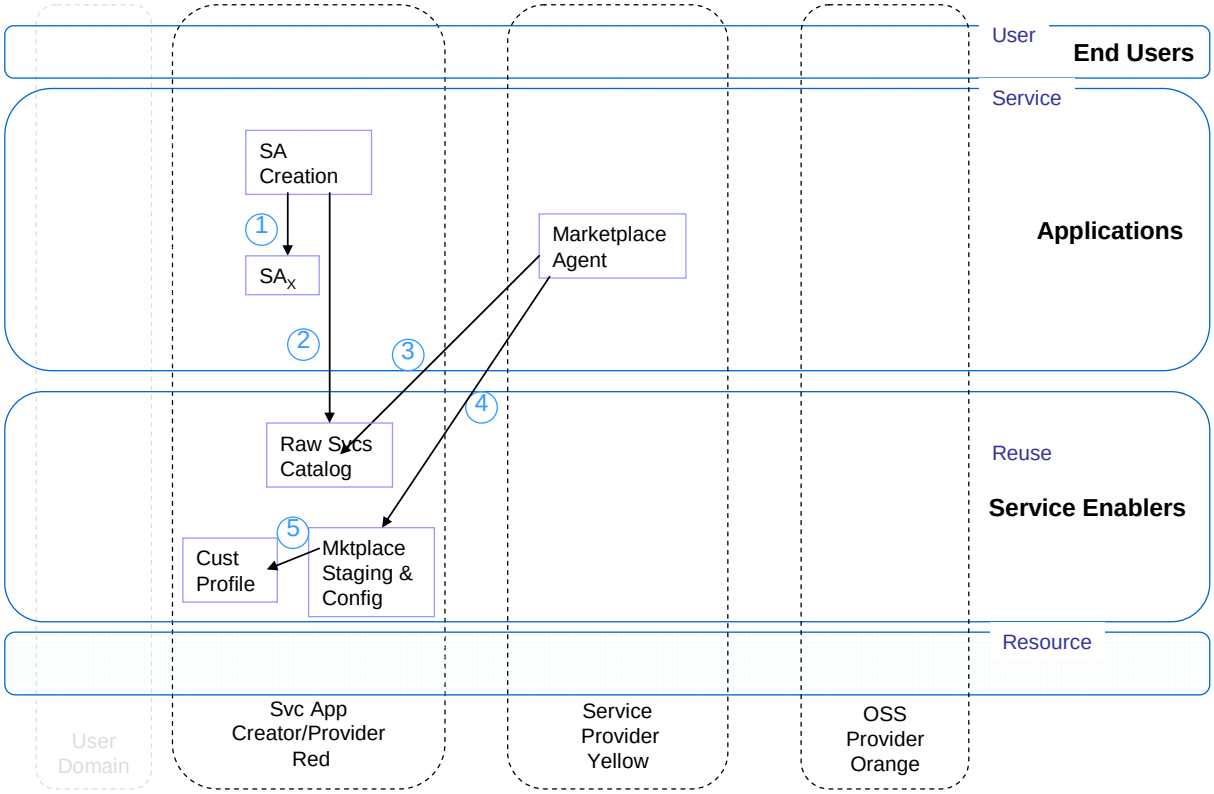


Figure 2-3 : Service Syndication

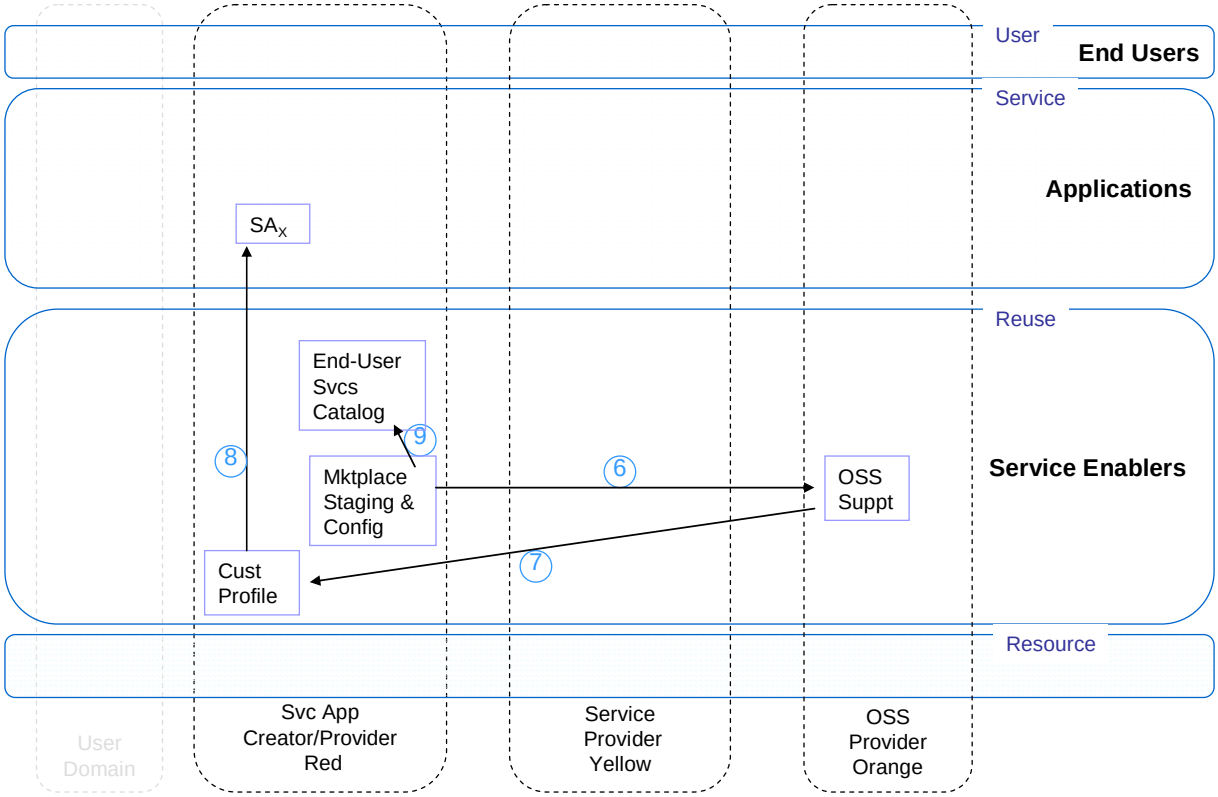


Figure 2-4 : Service Syndication – continuation of Figure 2-3

2.2.2.3 **Alternate – options**

Here are some example variations:

- The Services Marketplace could be external to SAC$_R$'s domain.

  Note that the Services Marketplace is an entity in its own right that involves certain functions, into addition to the service catalog, such as Identity Management for customers or potential customers, and customer profile & context management.

- The copy of SA$_X$ that SP$_Y$ uses and causes to be configured could conceivably reside within Yellow's domain and Yellow could be responsible for configuring it.

- Yellow could arrange directly with Orange for management or billing support, in which case Yellow would also have to indicate details to Red, and Red and Orange would to determine the specifics of their bindings.

2.2.2.4 **Analysis in context of SON framework and standardization**

SA creation would in part allow for making use of a known set of SEs. Another consideration of a SAC environment worth highlighting would be how it supports building advertising or other promotional material into an application.

Some areas that may be worth considering for standardization are:

a. What information about a SA (or raw end-user service) needs to be available to potential purchasers/renters, and how it is formatted. E.g., what it does, charging, runtime environment options, operations support environment options and interfaces, capacity, advertising/promotional material options, etc.
b. The nature of the purchasing interface, including reflecting whatever agreement needs to be made between the parties.
c. The nature of the configurable attributes associated with the app component model. And the configuration interface. These should include support for the SA's advertising options.

Note that we are *not* addressing the standardization of characteristics of a SAC platform (e.g., user or other interfaces or software structure). One would expect that a SA Creation platform to some degree would resemble a SE Delivery Platform.

2.2.3 Use Case – Service Provider to Service Provider Handoff

2.2.3.1 **Overview**

The purpose of this use case is to illustrate inter-domain interaction at the application level that we may want to consider standardizing to some degree, namely

- a service handoff interaction

- a content redirection interaction

Note that this use case as depicted does not involve Service Enablers (but see the discussion under Section 2.3.3.3 below).

Here is the tangible scenario:

Mary is currently watching streaming video over her wireless phone from her wireless Service Provider Green (SP_G). That service is provided in a context with various other features. Mary wants to transition watching the video to her home TV, whose service is provided by Service Provider Blue (SP_B). Therefore, a handoff must occur between the two SPs. Overall service control passes from SPG to SPB. The feature-surround that SP_B would normally provide to this user (Mary) in a non-handoff situation comes into play.

Assumptions:

- After the handoff SP_G no longer has any involvement in this session with Mary; i.e. the SP_G feature ecosystem associated with Mary's session is gone.

- The content is supplied by a 3rd party, Content Provider Yellow (CP_Y).
- SP_B has connectivity with and can talk directly to CP_Y. (If not, SP_B may have some alternative arrangement with SP_G to receive the media through SP_G.
- There is no temporary interval in which the video is seen on both the wireless phone and the IPTV.
- Charging is not specifically addressed. It could, e.g., happen 'naturally'. I.e., whatever rate plan Mary has with SP_G is in effect while Mary is watching the video through that SP, and whatever rate plan Mary has with SP_B is in effect while Mary is watching the video through that SP, and there is no charge for the handoff itself.

There are other considerations not addressed here, such as priority, lawful intercept or Quality of Service (QoS).

We also chose to illustrate a content-streaming scenario. We could have shown, e.g., a hand-off involving a multimedia (Session Initiation Protocol (SIP)-based) session between end-users. Additionally, we chose to show the current domain/device as invoking the handoff, as opposed to the target domain/device or perhaps even a third domain/device that is neither current nor target.

2.2.3.2 **Detailed steps**

The steps described are relatively undetailed. For example, they don't show exactly when the session ends with the wireless phone, or whether SP_B initiates the session redirection with CP_Y before or after SP_B initiates the session establishment with the user's IPTV. Furthermore, the details of the media path establishment are not discussed, and that media path is not represented in Figures 2-3 and 2-4.

1. Mary has a video session active on her wireless phone device, provided by SP_G. Mary triggers her wireless phone to initiate a transfer request to her home IPTV.

2. The wireless phone application originates a request to SP_G for transfer to an identified target device, her home IPTV.

    - Unique identifier/address for the IPTV endpoint.

3. The SA_G controlling her video and feature-surround receives the request. Using underlying resources, SP_G matches the target endpoint identifier with the appropriate SA_B that a handoff needs to occur with. (The 'magic' whereby SP_G can correlate {Mary's handoff request, Mary's user identifier, IPTV identifier/address} data to SA_B is left open. It may, for example, involve prior SP-SP operations type interaction/cooperation that loads such information into a database somewhere.)

4. SA_G sends a handoff request to SA_B.

    - Content provider ID for CP_Y
    - session ID known by CP_Y

- target device identifier/address (for Mary's home TV)
- user identifier for Mary that SA_B can also understand as such
- handoff identifier (that SA_B will use with SE_Y)

5. In parallel with step 4 above, SA_G sends a handoff notification to SE_Y.. (The purpose of this notification is to allow SE_Y to know that the forthcoming 'redirect' request from SP_B is authorized by SP_G.)

- Service provider ID for SP_B
- handoff identifier

6. Having received the handoff request, SA_B initiates a session with the target IPTV.

7. Mary is alerted of the incoming session request to the home TV device and accepts the request.

8. SA_B receives the request and initiates a session 'redirect' with the Content Provider using its SE_Y.

- SP_B identifier
- handoff ID (that was created by SP_G)
- IPTV session ID (that was known to SP_G and CP_Y)
- access point into the SP_B network
- media/coding information

9. SA_B invokes Mary's feature-surround for this type of session, based on Mary's user identifier. Such invocation may involve service enablers, other underlying resources, or other service applications.

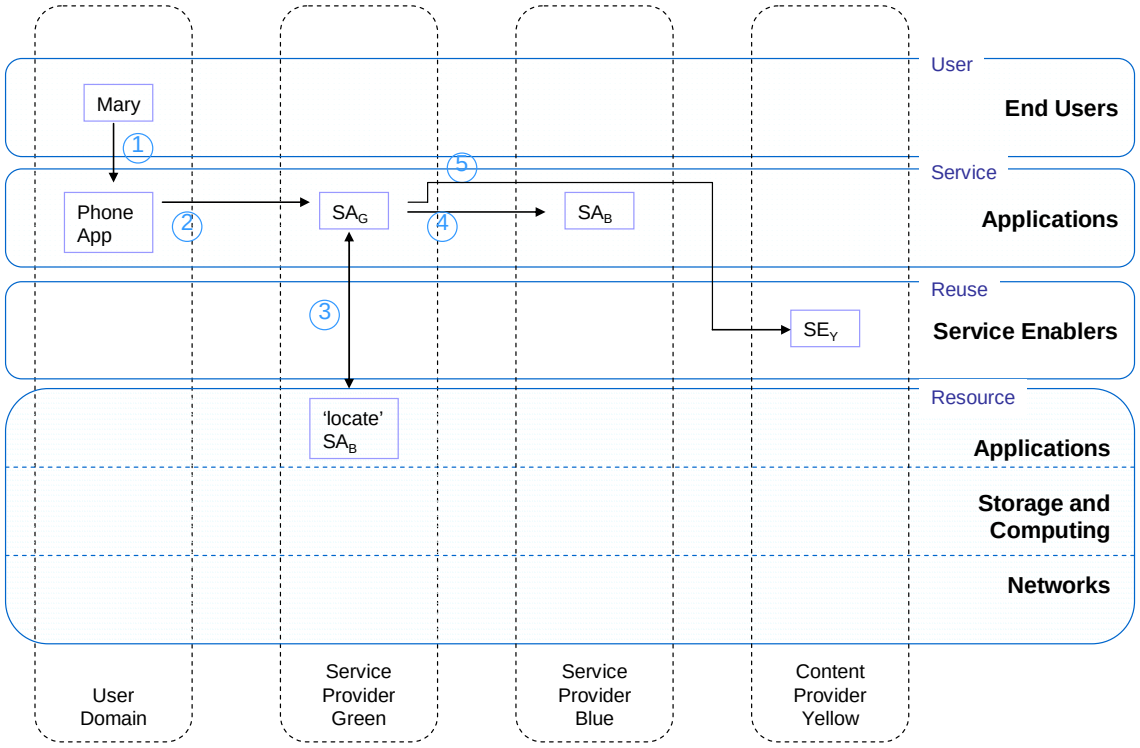


Figure 2-5: SP – SP Handoff

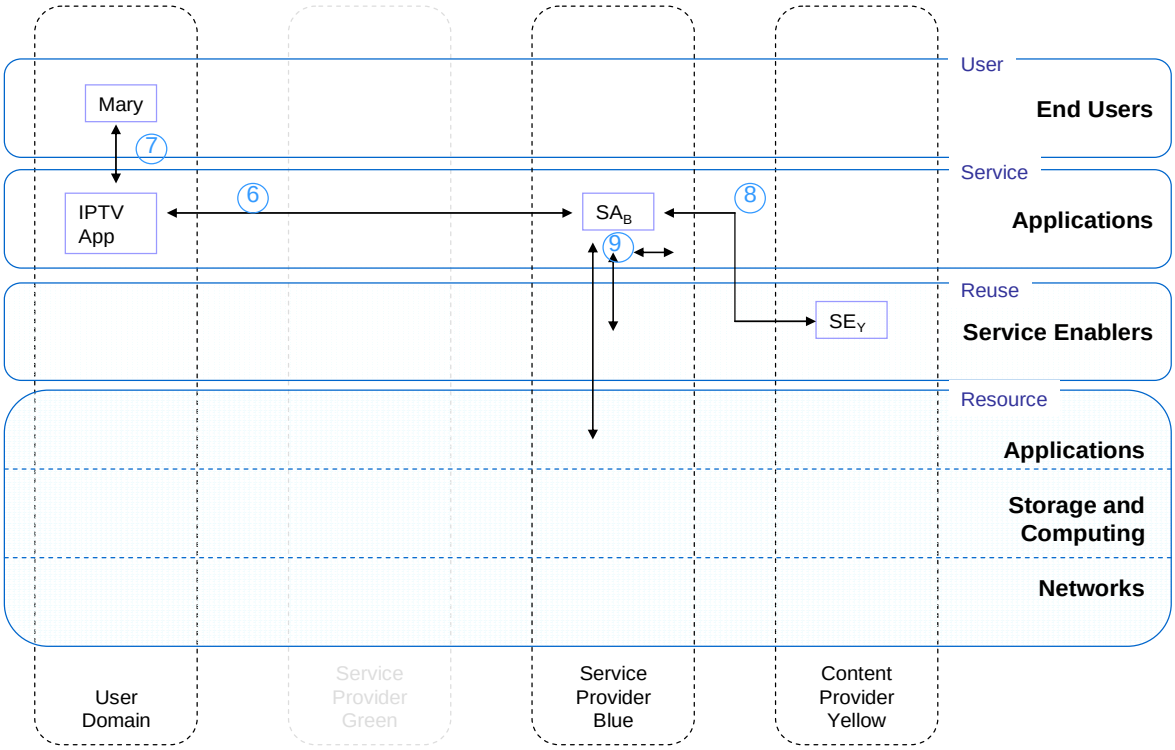


Figure 2-6: Continuation of Figure 2-5.

2.2.3.3 **Alternate – options**

Alternatives to the SA_G → SA_B handoff interface approach described above could be SA_G→SE_B→SA_B or SA_G→SE_G→SE_B→SA_B or SAR_G→SIM_G—SIM_B→SA_B or some other combination. I.e. the inter-domain relationship could be mediated by one or Service Enablers or Service Interaction Managers.

The SA_G→SE_B→SA_B alternative may be of especial interest. One could imagine that SP_B deploys a "handoff service enabler" that any other SP could be aware of and use to invoke a handoff to SP_B, without the handing-off SP having to know about any specific service applications within SP_B.

We have chosen to represent the content delivery functionality in Content Provider Y that interacts with a Service Provider as a Service Enabler – SE_Y, rather than as a Service Application, because we think that positioning it the SE level suffices and is preferable.

Additional scenarios of potential interest, but not detailed at present include:

a. Mary is watching the video session on her wireless phone while roaming and wants it transferred to another device. (Presumably if she's roaming with one device, she's roaming with both the transfer-from and transfer-to devices.)
b. Joe is watching a video over the internet and decides his friends Jill and Jack should watch it with him on whatever devices they can currently be reached at (Jill – wireless phone, Jack – Wi-Fi device). Joe initiates a request to invoke their inclusion.
c. Mary has been web browsing on her laptop and wants to transfer that current context and web-browsing history to his/her cell-phone so she can pick up on her cell-phone where she left off on her laptop, including, say, watching a video on U-Tube. This one may involve a solution where there is little/no network functionality involved and is driven by software in the current/target end-user devices.

2.2.3.4 **Analysis in context of SON Framework and standardization**

At the highest level, this use case simply highlights the need to commoditize inter-domain handoffs at the application level, however that is best achieved (direct application-application or through one or more service enablers or through one or more service interaction managers). The right technical solutions for implementing such a handoff requires further investigation, but the following considerations may generally apply.

- Application-level capabilities to support invoking and receiving a hand-off, both in terms of real-time cross-domain interfaces and application creation.
- Sharing across domains the context data related to status of the current 'feature surround'.
- Sharing of user profile data across domains.

A cross-domain capability to allow one domain (SP_A) to assert to a second domain (CP_Y) the trustworthiness of a third (SP_B) as pertains to a particular transaction.

A Service Enabler whose job it is to accept (or possibly initiate) a handoff request on behalf of arbitrary applications. In that case, perhaps there is 'application profile' data that needs to be shared with the SE so that it can pick the appropriate SA to receive the handoff.

This use case is an inter-domain use case. We understand that the intra-domain case is being addressed to some extent in the 3rd Generation Partnership Project (3GPP), especially a media session handoff among devices across wireless (cellular), wireline and Wi-Fi access within an IMS domain.[1]

2.2.4 Use Case – Global Presence

The representation of Service Oriented Networks within a two dimensional framework coupled with the nature of the Reuse layer within the framework creates a challenge in representing the reuse of Applications developed within the Services layer. In order to avoid unnecessarily complicating the representation of the framework, Applications in the Services layer which also serve as an Application within the Resource layer may be identified within a second independent use case.

The following use cases demonstrate a Global Presence Telco Orchestrated Service Application which is then offered for integration into a Social Networking Video Service.

The Global Presence Telco Orchestrated Service Application enables a mobile user to subscribe to a global presence service hosted by the fixed Telco. The global presence server maintains presence states for multiple Resource Applications with messaging into the Reuse layers for Short Message Service (SMS), IPTV, and IM Presence states. This Global Presence service:

- Is triggered through standard SUBSCRIBE and NOTIFY requests
- Leverages Resource Applications through Service Enablers within the Mobile Telco, Fixed Telco, and multiple (web based) 3rd parties.

This Telco may choose to further develop the Global Presence application to expose its functionality as a re-usable service enabler within the Reuse layer. In this use case, the Telco exposes the Global Presence functionality installed on an appropriately dimensioned presence server to third party application developers. In turn, the third party application developers integrate the use of global presence into a Social Networking Video Service facilitating video sharing and messaging among its users:

[1] TS 23.237 *IMS Service Continuity* (focused on same device inter-access)*; and especially TR 23.893Feasibility Study on Multimedia Session Continuity* (including session or media transfer from one device to another) leading to a new SA1/SA2 WID *IMS Service Continuity Enhancements: Service, Policy and Interactions* in draft form (see S2-085082); possibly other related WIDs – It is not clear whether there will be one or multiple (seems like Inter-Device Transfer is now in the same WID?) at the time of writing this note.

- Triggered through standard SUBSCRIBE and NOTIFY requests by the mobile user
- Leverages Global Presence as a Resource Application through Service Enablers within the Mobile Telco, Fixed Telco.

2.2.4.1 Use Case A

Sarah turns on her mobile phone to contact Jack. She looks up Jack in her contact list and can see that Jack is not available on his mobile phone but is available via IPTV and his social networking site. Remembering that Jack was watching the "big" game with his friends, Sarah decides to send a generic "good luck" message to Jack via IPTV and a more personal message via his social networking site.

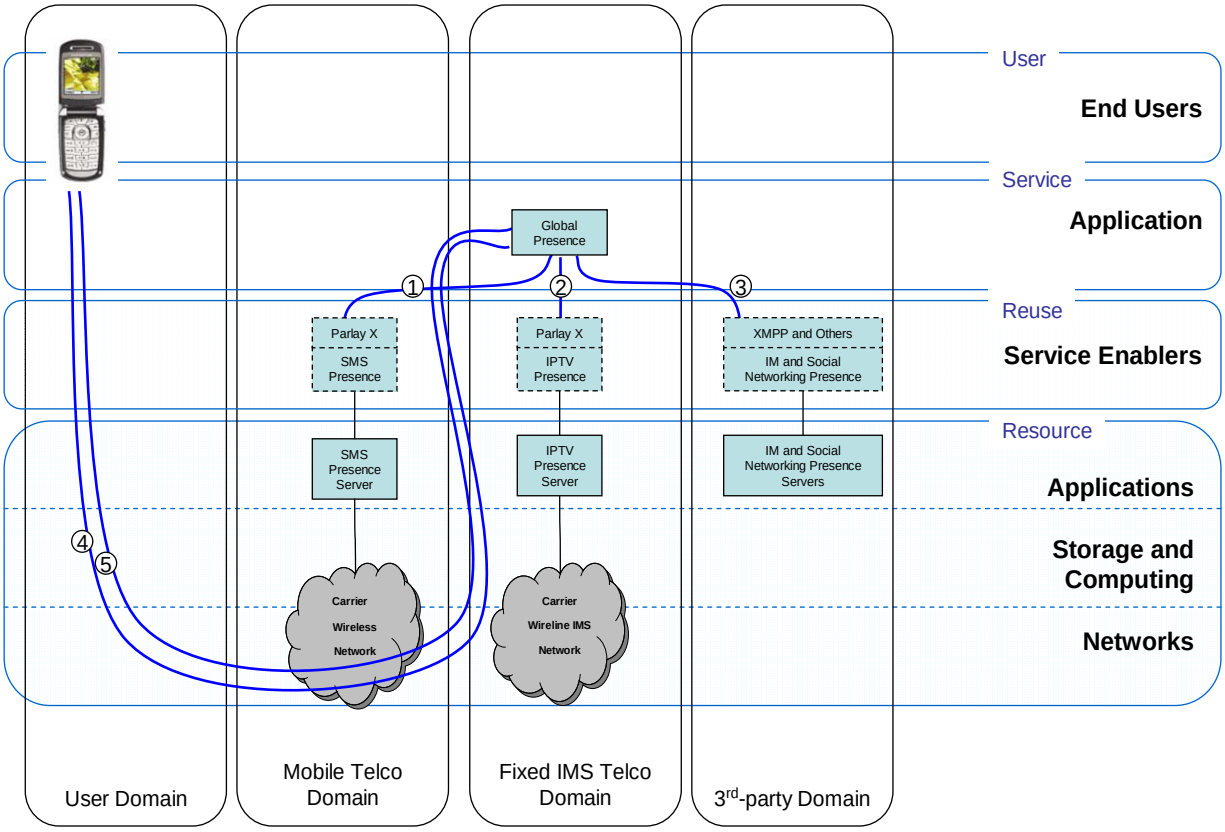


Figure 2-7: Global Presence Use Case

2.2.4.1.1 Detailed steps

1. The Global Presence Server subscribes to SMS Presence through startPresenceNotification and endPresenceNotification Parlay X requests. Upon a startPresenceNotification, the presence server will send an immediate NOTIFY in response and will send future NOTIFY messages as the underlying state of the presentity changes.

2. Similarly, the Global Presence Server subscribes to IPTV Presence through Parlay X.
3. The Global Presence Server subscribes to IM and Social Networking Presence through the appropriate protocol interface. One possible interface is through XMPP.
4. The Mobile user subscribes to the Global Presence Server through startPresenceNotification with an outgoing SUBSCRIBE request and receives an incoming 2xx response to the SUBSCRIBE.
5. The Mobile user receives a NOTIFY request from the Global Presence Server containing the presence information and responds with a 2xx to the Global Presence Server.

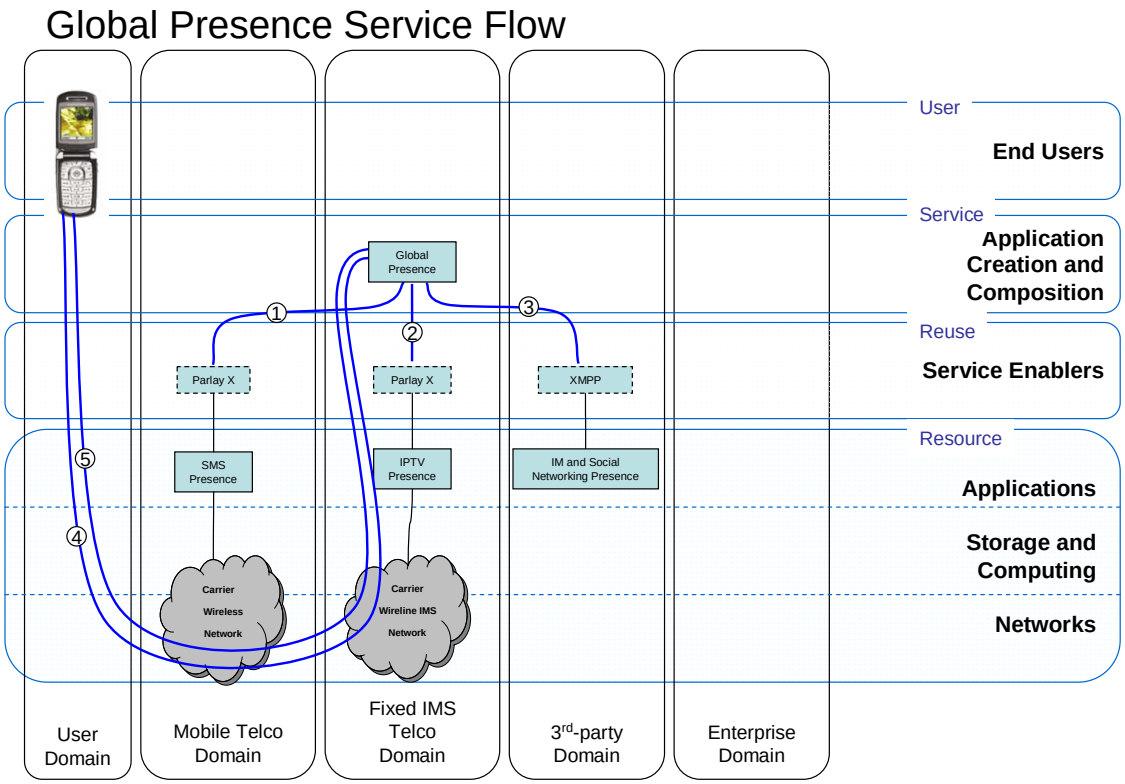


Figure 2-8: Global Presence Service Flow

2.2.4.2 **Usecase B**

Matt is a professional videographer who participates in several professional and social interest groups. Matt often sends out short segments of his work to these interest groups and has signed up for a social networking video service to increase his success rate in getting the content in front of his targeted audience. The social networking video service provides Matt and the distribution applications within the service presence information for interest groups.

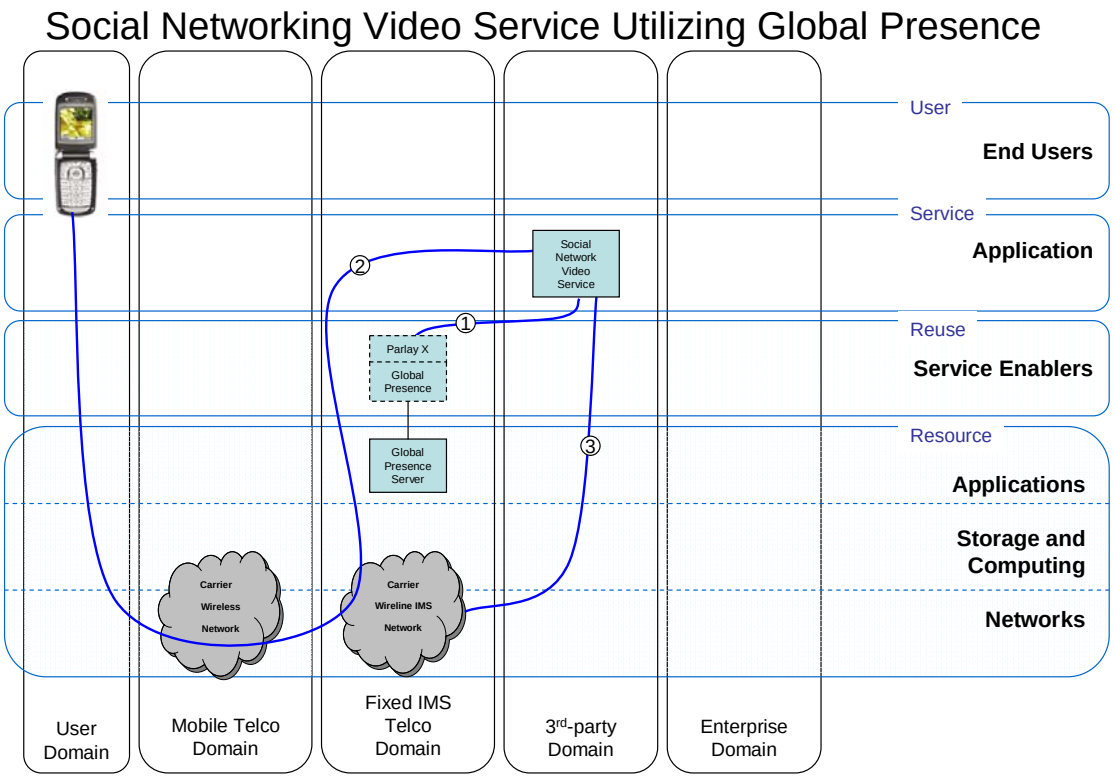


Figure 2-9: Global Presence as a Resource Application Use Case

2.2.4.2.1 Detailed Steps

1. The Social Networking Video Service subscribes to Global Presence through startPresenceNotification and endPresenceNotification Parlay X requests to the Global Presence Server. Upon a startPresenceNotification, the Global Presence server will send an immediate NOTIFY in response and will send future NOTIFY messages as the underlying state of the presentity changes.
2. The Mobile user submits a request to the Social Networking Video Service to share a video with another user in a location that they are currently available.
   - Potential charging event (based on user's broadband plan)
3. The Social Networking Video Service considers the receiving user's presence status and delivers the video.
   - Charging event (Video Service)
   - Charging event (Global Presence)

2.2.4.3 **Alternate – Options**

This Global Presence use case is intended to illustrate the methodology for using Service Applications as Applications within the Resource layer. Thus, this use case does not detail the subscribe and notify process by which the presence states for SMS, IPTV, and IM are maintained by the global presence server. The use cases can be easily modified or extended to include these specific message flows and to generate additional scenarios which include the usage of the presence information provided through the services.

2.2.4.4 **Analysis in context of SON framework and standardization**

The use case illustrates the flexibility of the SON framework to function in scenarios which cross functional domains and reuse applications in the Service layer as applications in the Resource layer. When creating applications within the Service layer, service enablers may be developed which treat the Service layer application as a Resource exposing functions within the application for reuse within or across domains. In order to provide clarity, it is recommended to represent the cyclical nature of the usage in two separate illustrations.

2.2.4.5 **Service Oriented Architecture**

Interoperable interfaces are required between the Service level applications and the service enablers within the Reuse layer of the framework and may be applied to intra and inter-domain usage. In particular:

1. Global Presence Application and the SMS Presence Service Enabler – Parlay X
2. Global Presence Application and the IPTV Presence Service Enabler – Parlay X
3. Global Presence Application and IM / Social Networking Presence Service Enablers –XMPP is one of several possibilities
4. Social Network Video Service and the Global Presence Service Enabler – Parlay X

2.2.5 Ad sponsored Location Based Navigation

This use case demonstrates a Telco Hosted/Orchestrated Service Application that:

- Is triggered through IMS
- Uses Service Enablers of a Mobile Telco and multiple (web based) 3rd party service enablers.

2.2.5.1 **Overview**

Roger uses his mobile phone to call a restaurant to make reservations and ask for directions. The restaurant offers to send the directions from Roger's location to his cell phone. Roger agrees and a short time later he receives a message with turn by turn directions followed by another message that contains an advertisement for a pub next door to the restaurant with buy one get one free offer.

2.2.5.2 Detailed steps

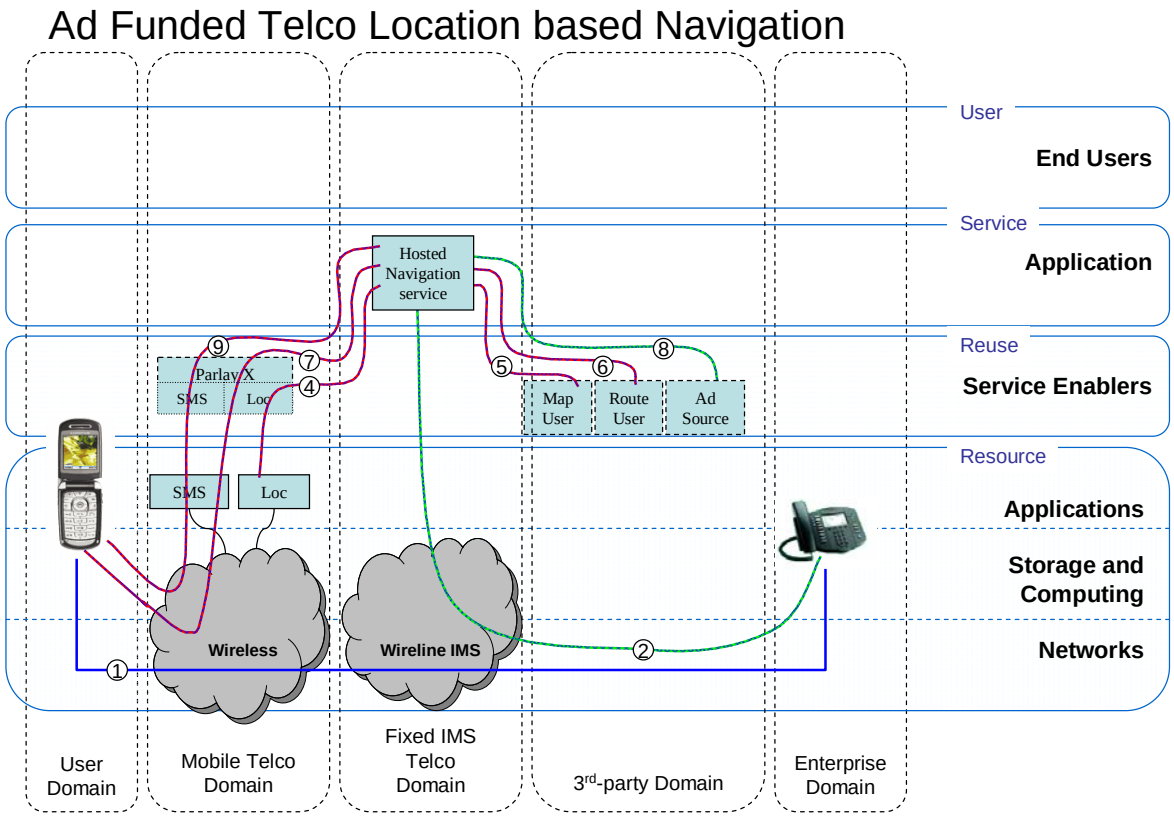


Figure 2-10: Navigation Use Case

1. Mobile user calls restaurant for reservations and directions.
   - Normal Call Control and Charging Apply
2. Enterprise initiates Location Based Navigation Application through IMS.
   - Charging event (Enterprise)
3. *(3rd party only) IMS-Application Server (AS) invokes 3rd party location based navigation application and delivers user Mobile Number and Business Street address.*
4. The Navigation Application invokes a Parlay-X terminal location Service to determine the current location coordinates of the mobile user location. (along with the users profile, preferences, and permissions??)
   - Charging event (Nav)
5. The Navigation Application invokes a 3rd Party Web Service to map the coordinates to a street address.
   - Charging event (Nav)
6. The Navigation Application invokes another 3rd Party Web Service to calculate the turn by turn driving directions between the mobile user location and the business facility street address.

- Charging event (Nav)

7. The Navigation Application invokes a Parlay-X Message (SMS) Service to send the directions to the Mobile Phone.
   - Charging event (Nav)
8. The Hosted Navigation Application invokes a 3rd Party Web Service to select the advertisement which could be based on the mobile user's current location, his destination (the restaurant) or some other profile or preference criteria.
   - Charging event (Ad)
9. The Hosted Navigation Application renders the advertisement / coupon to the end user in the same manner as the turn by turn directions were delivered.
   - Charging event (Nav)
10. The Navigation Application delivers charging events (Call Detail Record?) to billing center.

2.2.5.3 **Alternate - Options**

The use case is easily modified or extended to create additional scenarios. The Navigation service could be provided by a 3rd party and triggered from an IMS Application Server. Step 3 has been included in the detailed to indicate how this might be done. The service could be extended to include a short video of the restaurant, its specialty dishes, surrounding location or other topics of interest.

2.2.5.4 **Analysis in context of SON framework and standardization**

This use case demonstrates the capability of the SON framework to function across domains (in this case, Legacy Mobile, Fixed IMS, and Multiple IT based 3rd parties) to easily create new services without modifying the core of each of the domains.

The Use case also demonstrates the value of the SON framework in that the Telco can both leverage 3rd party service enablers to create new services, but can also expose service enablers to 3rd parties. The assumption is that both serve to generate new revenue opportunities.

2.2.5.4.1 Service Oriented architecture

Interoperable interfaces are needed between the Service Application and the Service Enablers. In this case this includes interfaces between:

1. Hosted Navigation Service Application – Map User Service Enabler
2. Hosted Navigation Service Application – Route User Service Enabler
3. Hosted Navigation Service Application – Ad Source Service Enabler
4. Hosted Navigation Service Application – Parlay X SMS Service Enabler
5. Hosted Navigation Service Application – Parlay X Location Service Enabler

In this particular example, the service application/service enabler interface is cross domain. The same interface is equally appropriate for intra-domain service application/service enabler interface.

2.2.6 Use Case – Advertisement

2.2.6.1 **Overview**

This use case enables targeted web advertisement for users of a mobile group communication service. The objective of the use case is to illustrate an application that combines user experience over a mobile network and over internet, using different service enablers, devices and user contexts.

User experience overview:

- User / groups of users subscribe to a mobile community service. They can do location-based group chats, or other form of communications relevant to their context.
- They get incentives on their communications services if they click through ads they receive during their web browsing experience. These ads are related to the context of their mobile community communications, like group location.

Actors and components (see Figure 2-11 below)

- An end user is both a subscriber to a mobile SP and a user of the web from another SP. The user domain typically contains resources and enablers in user control and ownership like Personal Computer (PC), Browser, other application components
- The mobile SP domain is purposely split into different domains to exemplify the inside as well as outside boundaries of global SP:
    - Network domain, e.g. owning the resources for network and media
    - IT domain, e.g. owning resources for subscriber management
    - Virtual SP is operating the mobile group communication service on top of enablers provided by the underlying mobile network.
    - The Location and Group calls service enablers rely on resources such as conferencing application, instant message and location servers
    - The application inside virtual SP uses a Subscriber mine enabler used for aggregating and sharing information on subscribers and groups
- The web and content application domain provides web content and can collect ads from some content provider based on contextual user data (note this domain separation is not shown explicitly)
- The mobile SP provides these contextual data in order to produce targeted ads, and then obtains ad clicks reports in order to sponsor subscriber's usage further

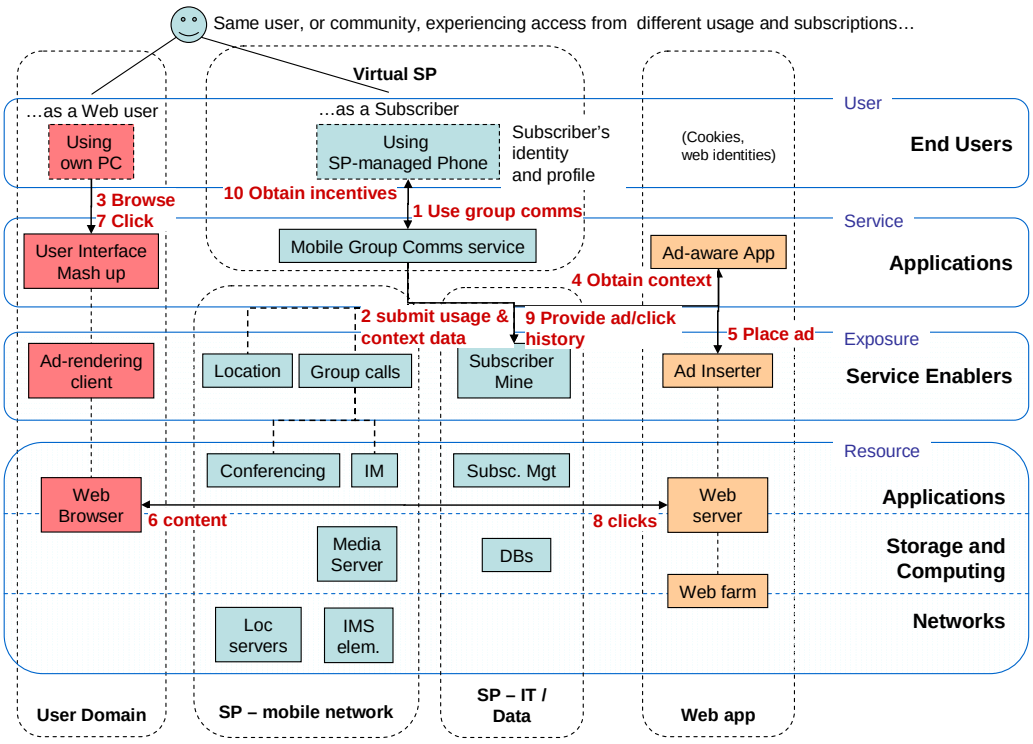


Figure 2-11: Web Based Advertisement

2.2.6.2 Detailed steps

1. Subscribers use group services to connect with location-based community, likely with an SP-managed phone that may contain client-side application components
2. SP captures location and other context or usage history data from this group application through a subscriber profile enabler (for further query)
3. user now browses web on the PC, or alternate device, likely not under SP control
4. the web app recognizes user, in a cross domain way, obtains context information for this user from the SP's subscriber data enabler
5. web app then requests targeted ads placement, possibly from a 3rd party content provider (collapsed in the picture)
6. customized content is sent to and rendered by the browser (can be dynamic mash up through e.g. an Ajax client). The ad is rendered in so that the end user knows it is SP sponsored (SP's logo placement by Web app may be based on another SP's enabler)
7. user clicks on an Ad and demonstrates some level of attention that the client component can check
8. the ad-click is communicated to the web application
9. the web application then reports to the SP that a subscriber has successfully clicked on a particular personalized ad

10. the SP can then decide to provide incentives / coupons … based on the ad history, for further motivating such clicks-through. The ad history is also used to bill the content / ad company

2.2.6.3 **Alternate – options**

SP-based: the SP can provide ads delivered through the Virtual SP services directly, e.g. on the mobile phone

- The Ads placement enabler is used by the SP's group communication service for inserting ad content inside the communication or shared content.
- The SP apps can still collect ad content from a 3rd party managed enabler

Client-based / Web 2.0: the SP can control the client-side enabler for collecting ads clicks and related information

- The SP controls the end user enabler "Ad rendering" in order to capture the clicks directly without going through the Web app. In that case the SP application controls user domain enablers.
- The SP may deploy an Ajax script in the browser when the user wants to benefit from the subsidized group communication service.
- This application can be assembled (mashed up) on demand by end users, selecting which domain of communities and ads they want to be associated with, independently of which SP or internet SP they use.

2.2.6.4 **Analysis in context of SON Framework and Standardization**

Service Oriented architecture

This use case shows a user experience distributed over several applications, domains and set of enablers. Some of the enablers rely on an IMS enabled mobile network reusing some of the underlying resources (e.g. conferencing, location).

Some components of the applications can be based on Web 2.0 browser environments, possibly under control of the SP as well. This will require ability to deploy such components with adequate level of trust (e.g. using device-side technologies).

Service Enablers and composition

Virtual SP's application requires access to the functional interface of the underlying enablers. It also needs to handle operational information, e.g. for provisioning new subscribers, thus requiring access to non-functional interfaces of the enablers.

The Subscriber mine is made an enabler so that it can be shared across such applications and used to aggregate users' data (profile, preferences…) spread across other services.

Interfaces and protocols used to access enablers may vary depending on the environment (e.g. on device, browser, phone, server sides), while providing a consistent level of functionality

Service Creation

In order to enable on-demand and rapid service creation, e.g. user-driven mash-up or retailer-driven for sponsoring a live event, it is possible to assemble enablers from different SPs without specific business relationships to be set up.

Such enablers will have to be discovered and some information will have to be shared so that composition can happen either on the server side or on the user domain without the needs for complex provisioning.

End user / customer environment

This use case requires "identification" of users across the web and mobile domains, so that context information can be retrieved (note: identities can remain private, just some attributes can be used).

The Web app needs to recognize the user as an SP's subscriber and relates to the appropriate enabler. This may require some interaction with the user at some point in time to provide authorization. This can rely on some automated policies or end users preferences.

Service delivery, management and business model

This use case exemplifies a diversity of business organizations in the service delivery chain while needing a consistent user experience to be delivered. One objective is to enable a rapid service creation even in lack of private business relationships.

This use case requires handling appropriate end to end multimedia QoS if the ad content is to be delivered through alternate means than pure web data (e.g. over a video channel). A Content provider may want to guarantee the ad click-through experience is of quality thus requiring end to end Quality of Experience (QoE) to be managed across the domains.

Considerations on existing applicable work in industry standardization, activities or gaps

- How to expose enablers to diverse application environments with a consistent level of functionality (i.e. diverse protocols and languages, device or server sides)
- How to correlate users and subscribers and discover their (home) SP and enablers
    - Federation, identity, service identifier and discovery across domains
- How to handle business aspects of reusing enablers across domains in a rapid and automated way
    - Interfaces for business relationship management at enabler level
    - How to handle trust in an end to end composition of enablers over different domains, without threatening privacy nor asking permission each time
    - Deployment of enablers, use and management of other domain enablers (includes simple provisioning policies, preferences, etc)

- How to deal with end to end cross domain QoE so that one SP actor can guarantee the user experience expected from other actors / enablers
    - Across multiple devices, multiple enablers…
    - Define, manage, notify QoE properties in an automated way

3 SERVICE ORIENTED NETWORK ARCHITECTURES

3.1 Perspective on IMS

Any comparison of IMS, SOA and Web 2.0 must first acknowledge that IMS has been designed and developed through an Industry Standardization body: 3GPP, which is now driving future IMS standardization work, commonly adopted by 3GPP2 TISPAN ATIS and International Telecommunication Union Telecommunications Standardization Sector (ITU-T) NGNs, by addressing their additional modes of access (wireline, Wi-Fi, evolved cellular …). While SOA and Web 2.0 have emerged as a de-facto industry "standards" there is no formal 'Standardization Body' that is responsible for driving these two initiatives (though they leverage a set of technology standardization efforts), and hence while the concepts may be well understood there is a level of flexibility in their definition and implementation that isn't present in IMS.

IMS reflects the principles of the service oriented architecture concept in that it is decomposed into a set of well-defined functional entities with a diversity of well-defined interfaces. They however operate in a more tightly controlled manner for delivering some of the core services of IMS like multimedia telephony. Other applications can be added on top of interfaces exposed through IMS application servers who can then participate in the control or coordination of IMS sessions and functions (enablers). IMS was though mostly designed with the network in mind and less from a service and user experience creation platform.

IMS, at its core, defines a network layer (Transport and Control Planes) and a service layer (Service Plane). IMS services can be developed (composed) using IMS service enablers linked together in sequence (a SOA concept) through a service workflow. Whether this sequencing is implemented using the IMS Service Capability Interaction Manager (SCIM) ('lower level') or at a 'higher level' using tools such as Business Process Execution Language (BPEL) is typically left up to the service developer's discretion. Hence it can easily be argued that IMS "utilizes SOA concepts"; however IMS is not an instantiation of SOA – since some IMS services are tightly-coupled with underlying network layer functions.

The well-defined interfaces of IMS focus on how functions interact together in a very well specified manner, rather than for these functions to expose an interface so that they can be orchestrated or choreographed into a coordinated manner without prior knowledge of the service context. An SOA approach would typically separate this composition or orchestration

of underlying services from their own service definition (and contracts). To further the comparison one could envision that an IMS telephony service could have been described as an SOA service orchestration script, instead of well-defined call flows.

The IMS approach has however resulted in a very robust IMS framework for dealing with trusted behaviors, call flows and hence guaranteeing the serviceability of the infrastructure and quality of service to the end users. Most importantly the ability to know the end user subscribers and manage the profiles to render services appropriately needs to be leveraged and kept as an important property for cross-domain SON.

3.2 Perspective on SOA

SOA utilizes loosely-coupled service data exchange to implement business processes. These business processes may be complex service value-chains such as Order Management or more simple service workflows such as an SMS-based Stock Quote service. IMS Service plane service enablers can be integrated into rich-communications service flows by simply extending an SMS-based stock quote service to include a "Call Broker" capability. This extensibility is at the core of a SOA design approach, where existing services can also be considered as potential 'service enablers' and can be re-used to create more complex, feature-rich services.

Service enablers (or service producers) are registered within a Service Registry and can be discovered by service consumers. The Service Registry contains service binding references (service contracts) that enable the service consumer to determine the data exchange structures and the actual service access end-points. A typical SOA Service Registry instantiation is implemented using an OASIS UDDI (Universal Description, Discovery and Integration) deployment.

Adopting a SOA design approach does not mandate that Web Services, and by association Web Service Description Language definitions (WSDLs), must be defined and consumed in the business process flow; however the loose-coupling and data exchange capability that Web Services enable has resulted in their adoption as the basis for many SOA service implementations. The fact that Web Services may be embedded in complex service flows defined in BPEL, and that BPEL service flows in themselves may be defined as Web Services is another example of how Web Services have become synonymous with SOA.

Since SOA utilizes a loosely-coupled design methodology the need for service management (or governance) has emerged as a key requirement. While a service enabler can be defined though a specific Web Service WSDL instance registered in the SOA Service Registry, it is expected that this service definition will evolve over time and that today's services may become tomorrow's service enablers. This capability requires that a structured service management function be applied to all service definitions to ensure that when a service enabler (producer) is modified that the appropriate service consumers are alerted to the change and invoke the correct service definition version.

This SOA approach, whether instantiated through Web Services or other technologies (including integration with Web 2.0) is highly relevant to SON Service Enablers and composition. SON standardization must address cross-domain aspects that would not be covered adequately by existing standards.

3.3 Perspective on Web 2.0

Web 2.0 has emerged as a new service paradigm gaining significant market traction over the past 24 months (W3C's Semantic Web is now considered to be Web 3.0). Web 2.0 also heavily utilizes the concept of "services" together with light-weight business models (SOA's loosely-coupled data exchange interactions); however Web 2.0 services (or widgets) are typically designed using a REpresentational State Transfer (REST) programming model principle for access to data and functions – Web 2.0 services can also be developed using the SOAP programming model but this normally involves the use of more complex, elaborate XML message exchange. REST uses Hypertext Transfer Protocol (HTTP) interactions with XML or JavaScript Object Notation (JSON - a lightweight data-interchange format) payloads.

It is usually accepted that Web 2.0 principles cover (in addition to the web model):

- user or group generated content
- community effect to create, share, rate, evolve and adopt services and content at web speed
- ability to mashup, compose highly customized user experience reusing content sources and services APIs over the Web. The mashup can happen on the device or server side
- continuous beta, meaning rapid and iterative creation and tests of new apps and services, as well as involvement of end users in service creation and refinement
- indirect, if not unproven, business models that few companies manage to succeed on the long run

Hence content and user experience creation are key aspect of Web 2.0, along with experimentation of business models and fast learning curves as well as dying cycles (succeed or fail fast). One fundamental approach is that trust is often not provided by a subscription relationship or controlled authentication but through a community effect. "Being a member" may have more value in Web 2.0 than "Having the rights". Reconciling these models over cross domain application will need to leverage both the rights that Subscribers have but also the authority that they nurture through community memberships.

3.4 Comparison of Existing Systems

From this discussion the following table is looking at IMS SOA and Web 2.0 through the angle of a few principles that help with the comparison. However these may apply very differently in

each domain and hence this shall be interpreted as an illustrative view on how each domain has predominantly been designed. This is not an attempt to provide any rating or hard comparison among them, but rather informed input for the SON exploration. The additional comments provided often exemplify how SON principles must leverage the best of all worlds and apply it consistently across domains to meet the SON requirements.

	IMS	*SOA*	*Web 2.0*	*Comment in SON context*
Design principles and approaches	**Service experience** Designed with operator in mind, resulting in network and subscriber control. Core network and interfaces matter	**Application experience** Designed with Developer in mind resulting in SW engineering principles applied to flexible services composition: modularization, reuse, cooperation, multi-tiers… IT enablers matter	**User experience** Designed with User and communities in mind for rapid creation and adoption. Edge and devices matter	3 are needed! Offering successful services requires ability to deliver robust and interoperable applications that innovate user experience and leverage multi-vendor multi-domain flexible composition of service enablers
	Vertical (domain) integration where protocols predominate	**Horizontal framework** where Service APIs predominate	**User experience mash-up**. A service is designed by leveraging the predominant cloud of web tools, scripts, content … on-demand	Leverage service enablement across domains (e.g. IMS exposure) with ability to use rapid creation tools
	Designed by standardization, designed for and made to work through multi domains interoperability, delivered to users once proven, services figured out late (from stds to service)	**Enabled by open IT SW standards**, iterative development of applications composed with exposed services (from SW to service) . Some level of interop achieved mostly inside a single domain	**Leverage open technologies (software) and core web standards**. Experiment and refine services, open to get adoption, extend by implementation, refine standards a-posteriori (from	SON standardization needs to leverage the rising forces of implementers and communities to get it adopted more rapidly. Especially enabling ability to try services assembly rapidly and on –demand,

	IMS	*SOA*	*Web 2.0*	*Comment in SON context*
			adoption to standards). Often led by innovation inside a single domain without interoperability requirement	while keeping strong interoperability goals across domains
Service experience focus, or famous for (value = refer to laws[2])	**Connecting**, robustness One-size-fits-all Communications services, some content services. Services not defined by end users Value = network effect = $(N)^2$	**Programming**, reuse Flexible and adaptive business services composition through rapid discovery assembly, SaaS… Value = computing power + software productivity	**Creation**, innovation End users customize, generate content, tailor long tail applications, share with Communities and collaborate, improve … Value = community effect = $(2)^N$	Goal of SON is to speed up creation of services that leverage the network's robustness and differentiate through application programming It is important for SON standards to take into account the importance of the "content" and not only of the "control" channel.
	Device-to-device Session interactions	**Server-to-server Service interactions**	**Client-to-server Browser interactions**	REST principles can apply to server-server interactions. Concept of Session shall apply cross domain over SON, as well as needs for Service enabler to service enabler interactions
Illustrative Service Delivery	Telco grade **QoS**, real-time, low	IT grade **scalability** ,	Web grade, continuous **beta**,	Reconcile QoE for cross-domain

[2] http://en.wikipedia.org/wiki/Reed's_law, http://en.wikipedia.org/wiki/Metcalfe%27s_law, http://en.wikipedia.org/wiki/Moore's_law , http://en.wikipedia.org/wiki/Wirth%27s_law

	IMS	*SOA*	*Web 2.0*	*Comment in SON context*
requirements	delays, 100Kthings/sec, available and dependable	100Kthings/day, backed up	best effort, up to Millions hits/day	SON application delivery. Refine standards for SW programming to take into account QoS at application level
Usual Business models and monetization	Direct **subscriber** relationship, service usage, support	Application **Transactions and 3rd parties relationships**	**Indirect** models, no support, often cheap / free for end users	Ability for Son standards to enable innovation of business models is important. Componentizing service creation and delivery task enables players to foster innovation. Ability to know end user / subscriber needs to be open and enabled across the domains (e.g. re-aggregate profiles)
Domain Trust model	Built-in security model. In multi SP domains: subscribers can connect or roam. "Having a subscription" provides trust	Leverages IT security models (e.g. WS, Id, Policy), 3rd partners can combine or provide services. "Being a partner"	Fragmented models (e.g. Web security, web Id, community rules…), anyone can use. "Being a community member" leads to some trust	Sharing enablers across SON domains and SON users create challenges for authentication, privacy, profiles … standards need to apply and reconcile trust models for SON applications
Service creation and deployment environment	Intra-domain AS, internal SDP, IMS policies and interaction elements …	Orchestrating AS / SDP Inter service assembly	Browser as an execution container, SDKs, mashup APIs / GUIs	abstract protocol worlds into mashup APIs
	Services built bottom up from the network	Services built horizontally from SW reuse	Services built top down from the user	SON must foster reuse and user experience driven service creation

	IMS	*SOA*	*Web 2.0*	*Comment in SON context*
	Developers need to be telecom specialist to write apps, and learn protocols and call flows	State of the art SW engineering (object and service orientation...). Application designed through Web Services then deployed on multi-tier architectures (like SDP)	Developers need to be user experience specialists, often work with communities. Web tools and gadgets are leveraged on the edge along with the Web architecture	SON standards need to enable design of services and user experience independently of which service enablers are relevant Service assembly needs to work cross domain, service enablers need to be part of the cloud toolbox, hence providing single domain SDK is not enough
Typically famous for these technologies	SIP, Parlay RTP	Web Services, BPEL, XML	REST, Ajax, RSS,	
SP organizational focus	Network department.	OSS / BSS	IT and Web department	SON must leverage network and IT

Table 3-1: Comparison of Existing Systems

3.5 Gap Analysis

In order to enable widespread use of SON, standards need to take into account the combination of key principles of service creation from the environments described previously. That can be illustrated as follows:

1. user experience matters
    - Content and collaboration service components enrich user experience on top of communications and session services.
2. service deployment and operation is critical to establish new business models,
    - Interoperable service enablers bring strong deployment and operation properties in order to guarantee quality of user experience across domains end 2 end
    - IMS brings interoperable roaming, authenticated and mobile multi-media calls, subscriber, network and device information.

3. SOA principles to ensure strong SW principles lead to maximized quality / cost
    - SOA oriented composition and orchestration enables SW-efficient assembly and creation of services. Service Delivery Platform (SDP) / SOA brings ability to open networks to Web 2.0, enabling flexible reuse of service capabilities.

A schematic of how IMS, SOA and Web 2.0 standardization efforts can be projected, as of today, in a SON framework together is shown below. This illustrates how they influence the industry today, and projects how SON should extend their reach across the domains to meet the 3 principles above.

In the context of this figure IMS shall be understood as the most illustrative standards structure the industry has defined in the Telco domain that meets the needs for real-time applications. This is not restrictive of the capabilities that a Telco domain would typically use for delivering service applications. The intent is to stress the standardization divide or silos in a simplistic manner. Most importantly this is highlighting the limited connections these standardization efforts have today, both in terms of technologies and industry actors' involvement.

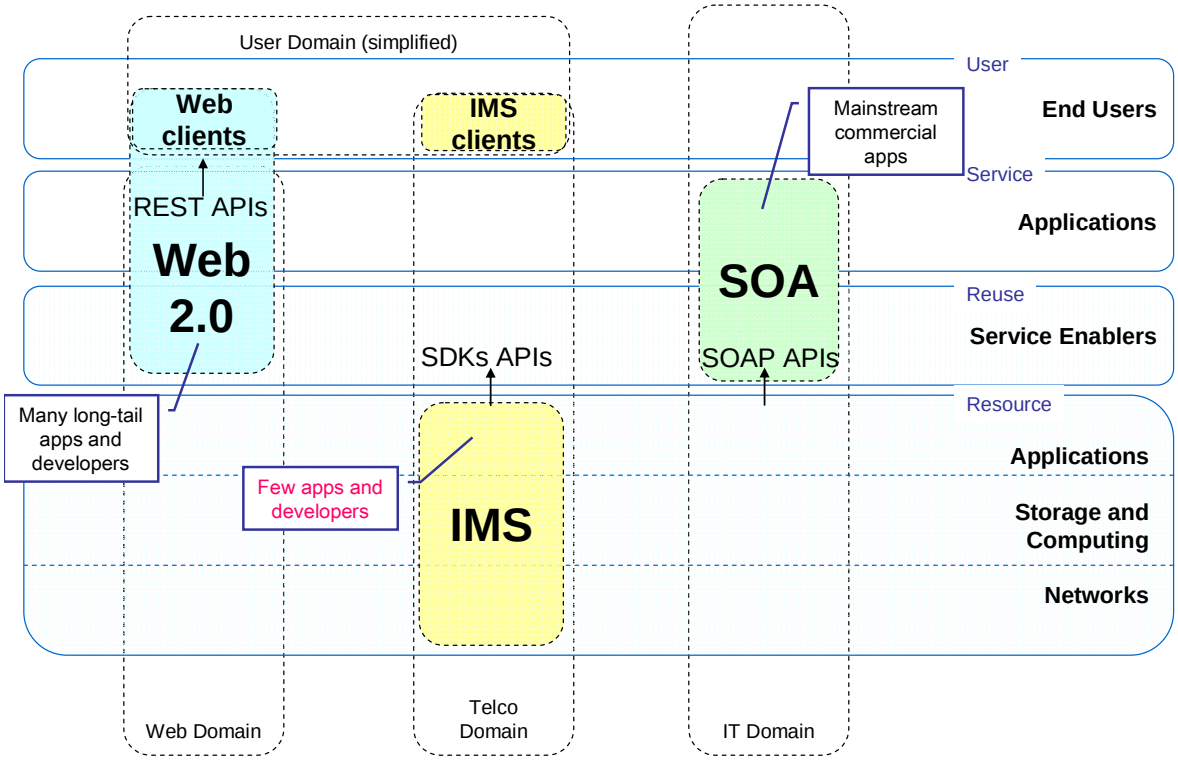


Figure 3-1: Perspective on today's focus of Standardization domains

Typically Web 2.0 and IMS can be seen as "stacks" to interconnect domains with user environment while the SOA can be seen as the application hosting environment in the IT /

Enterprise domain. Web 2.0 standards are typically W3C + REST architecture mostly used in the internet domain with REST APIs exposed to devices. IMS is only used in the Telco domain and has some client dependencies as well. SOA standards have been driven by major IT / Enterprise application requirements.

The next figure then projects the perspective for a SON Service Provider domain whose goal is to deliver applications that compose cross-domain service enablers, or expose service enablers to other domains.

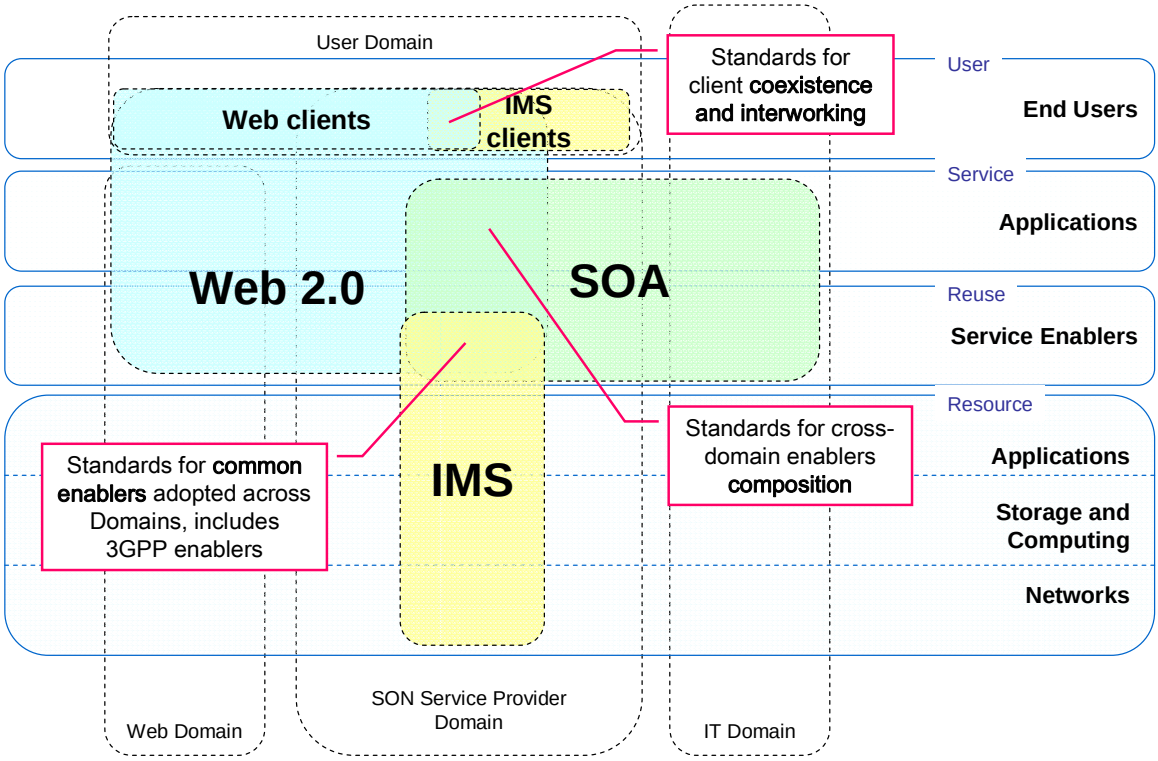


Figure 3-2: Perspective on SOA IMS Web 2.0 Standardization in context of SON Service Provider domain

Today, some existing Service Provider APIs already expose IMS Service Layer and SDP service enablers as RESTful APIs, enabling "Call Me" widgets to be embedded in web-pages or mashed-up with other Web 2.0 content feeds such as mapping services to deliver more interactive end-user services. Commercial off-the shelf (COTS) Application Server vendors are also offering Parlay X service enablers as RESTful APIs. That drives the needs for stretching standardization across domains and enabling better coexistence and convergence of the environments, including on the user domain.

One example of what SON standardization needs to address is how to operate the Web 2.0 stack along with the IMS stack from the same User environment while still leveraging the same trust model, especially when the service enablers are spread across domains.

Similarly, composition must be possible across technologies used for exposing enabler capabilities. For instance, functions in the IMS service layer are exposed as SOA components, and are registered in a SOA Service Registry as IMS Web Services (WSDLs). The IMS SIP Application Server exposes SIP Servlets (JSR 116 and JSR 289), and the IMS Parlay Application Server exposes Parlay X Web Services (or RESTful Web 2.0 services).

In this context, SOA can be viewed as the over-arching service composition model enabling services to be discovered, bound and consumed in services flows, where service flows are typically implemented in BPEL or another Business Processing Language. Again, SOA does not mandate the use of BPEL however; complex service value-chains that may exist for a period of hours, or days require a more robust, rich processing language.

Web 2.0 can be viewed as another method for designing and composing services that leverages SOA design concepts but utilizes the simpler REST programming model rather than the more complex BPEL processing language. Especially, Web 2.0 provides the connection to the user environment.

Finally, key common enablers should be identified across environments and networks and ensure interoperability across application domains and silos. Examples are identification/authentication, profile and charging / billing. Some of these enablers may come from service provider resources.

From a standardization perspective this convergence should also ensure the right industry actors are gathered together as it is as much important to have to right involvement across domains as to leverage the technologies.

The two figures shown above highlight the changes needed by traditional communication services providers (CSPs) and their suppliers in order for them to properly participate in the standardization of a service-oriented network. Traditional CSPs currently spend as much as five times more on the standardization of NGN and IMS than on SOA and Web2.0. CSPs and their suppliers must adjust the profile of their standards activities more in favor of SOA and Web2.0 in order to make SON a reality.

In doing so, CSPs will be able to adopt a leadership position, bringing together the worlds of IMS, IT and Web to create inter-domain, inter-operable standards that allow the whole market for re-usable services to grow.

4 SERVICE CREATION, DELIVERY, AND MANAGEMENT

4.1 Service Delivery

Service delivery refers to the way in which service providers create, deploy, operate and maintain services. Traditionally, this has been undertaken in the form of product or service stovepipes in which the whole service environment is self-contained. These stovepipes were custom-made to deliver against the specific product requirements, often without reference to other similar structures both within the service provider and externally.

With the advent of service-oriented architectures applied to communication service providers (CSPs) in the form of service-oriented networks, the requirement for delivering products and services concentrates on more open environments in which the services are constructed from component parts. Section 3.1.4 on agile service creation describes the way in which these services are created by developing and deploying applications based around the integration of service enablers. These service enablers are exposed using open, standard APIs with the intention that they are re-used by as many service applications as possible. A typical example is the function of authentication which is common to many services. An authentication service enabler would be created and deployed, and any service wishing to use this function would integrate the function of the service enabler into the service application.

With the advent of SOA, the industry turned its attention to describing a coherent environment in which services can be created and deployed in this way. The term SDP was used to describe a homogeneous platform that service providers could procure for this purpose. IT vendors proposed a number of different variants of an SDP with the following common functions present in most offerings:

1. The ability to create and deploy service enablers which provide the re-usable component building blocks for products and services. These service enablers would provide functions that represent the capabilities of networks, storage, computer processing, and software functions. Service enabler functionality would be exposed using open, standard interfaces, such as those provided by Web Services technologies (see section 3.1.4). Service enablers would be deployed in computing environments that contained additional software for their maintenance.
2. The ability to create and deploy service applications which may have been created at least in part by integrating the functionality of the exposed service enablers. These applications would present an interface to the user environment for the usage of the services they offered. Service applications would be deployed in computing environments that contained additional software for their maintenance. Service applications could be deployed in a service provider's domain, or they could be made available through a services marketplace for syndication to/use by other service providers.
3. An operational surround that provided a consistent way of managing the platform, the components deployed within it, and the services offered by it.

With vendors offering different types of SDP, a requirement for a definition and standardization has become clear and the TeleManagement Forum has established a program of work to begin this process.

4.1.1 Service Delivery Framework

In 2006 the TeleManagement Forum (TMF) began a study of service delivery with a view to developing a standard approach. Early work[3] from this study provided a framework which outlined the functional elements of service delivery and their interfaces to the network and IT resources; the user environment; customers, partners and suppliers; and OSS/BSS. A diagrammatic representation of this is shown in Figure 4-1, and comprises an operational environment supporting the following:

1. Service enablers as the building block for service applications
2. A means of exposure of components (primarily service enablers)
3. A means of network (and IT resource) abstraction to allow this functionality to be represented as a service enablers
4. A means of orchestration as part of the integration of service enablers into applications
5. The service applications themselves
6. A service registry

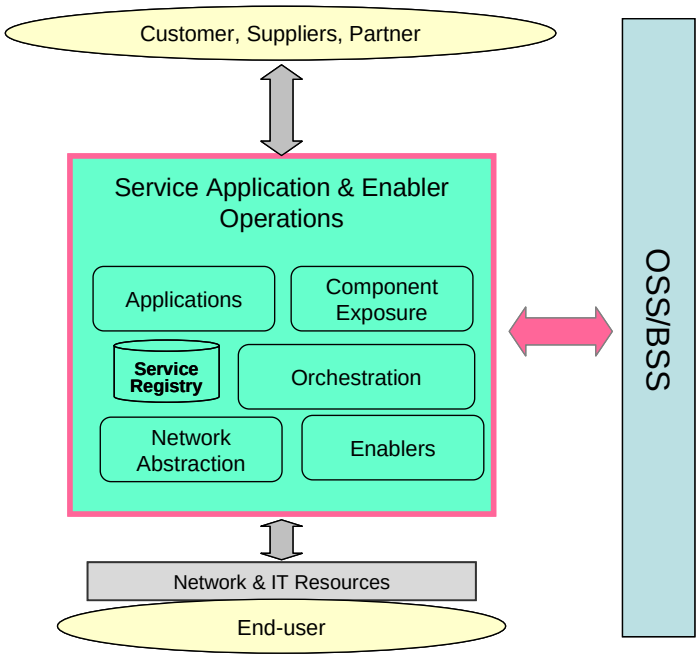


Figure 4-1: Service Delivery Functions

[3] Figures 1 and 2 do not represent official TMF output, but is nevertheless highly instructive.

In addition, this early work identified an outline of the service lifecycle that describes the way in which a service progresses from concept to market and during its operational lifetime. The service lifecycle is defined in terms of its overall management (and therefore contains many functions that are present in OSS/BSS) and is shown in Figure 4-2.

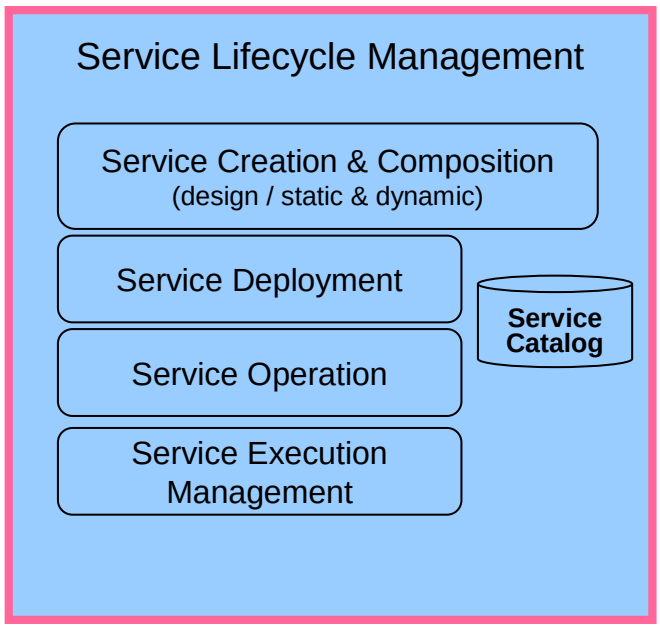


Figure 4-2: Service Lifecycle Management

The service lifecycle supports the following:

- Service creation and composition in which a development environment is used to code the service application including the way in which it makes use of service enablers.
- Service deployment in which a service application is taken from an environment in which it is constructed to one in which it is tested and verified to one in which it is deployed to users.
- Service operation which provides the elements to operate the service in use.
- Service execution management with which the general execution of the service to users is managed.
- The data which supports these aspects of the service are provided in a service catalog.
- Testing (although not explicitly shown) is a key element of each of the lifecycle stages.

The work of the TeleManagement Forum on service delivery has since matured into a major program of work on Service Delivery Framework (SDF), concentrating on the management aspects of service delivery. This is described in section 8.1.7, and includes the concept of an 'SDF Service' in the SDF reference model.

An SDF Service is directly analogous to a service enabler (as described in section ?) and supports a number of interfaces, as illustrated in figure 2; and particular must support specific SDF Service "Management Interfaces" that will be defined by the TMF These also include interfaces for its usage or composition with other services – the SDF Service Functional Interface. An SDF Service may itself rely on capabilities exposed by other services, such as those provided by an integration infrastructure, by network or IT resources, or by other SDF Services. The consumption of such external capabilities by the SDF Service is graphically represented by a consumer (a "half moon").

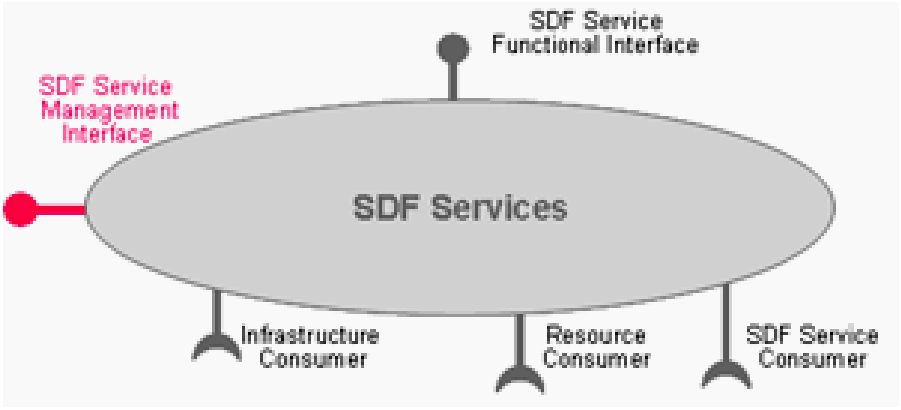


Figure 4-3: SDF Service

This highlights the recursive nature of the use of service enablers. Any service enabler can make use of resources to support its functionality – including that of other service enablers. Using this as a model for the overall composition of a service, one can consider the final service application to be a specialization of an SDF service in which the service functional interface is essentially the end-user interface of the service.

The following sections provide some illustrations of how service delivery maps inside the SON framework for the sake of identifying the key elements and standardization aspects. A number of use cases are used to show the different ways in which a service delivery platform can be used: 1) in a single domain application; 2) to support the deployment and exposure of service enablers; and 3) to support the deployment of applications and service enablers across multiple domains. The use cases also illustrate the role of the Service Creation Environment (SCE) (see section 4.1.2) in providing the entities that are deployed for service delivery.

Note: These examples and the use cases that accompany them, illustrate only a sub-set of the uses of a service delivery platform. The lack of industry consensus on the way in which SDPs are applied by service providers highlights the potential for different use cases.

4.1.2 Service Delivery to Support a Single Domain Application

This use case, shown in Figure 4-4, illustrates deployment of an SDP for single domain applications not using enablers, but combining resources together into a manageable application exposed to end users. This may use an SOA approach within a stovepipe application, or set of applications. Such an SDP deployment allows interaction with controlled user domain / user experience over well defined and domain-controlled interfaces. An example is a web application or a model of IMS applications which are deployed inside a single domain with full control over the handset environment. Note: the SCE is mostly addressing service creation by the service provider, utilizing domain-managed resources.

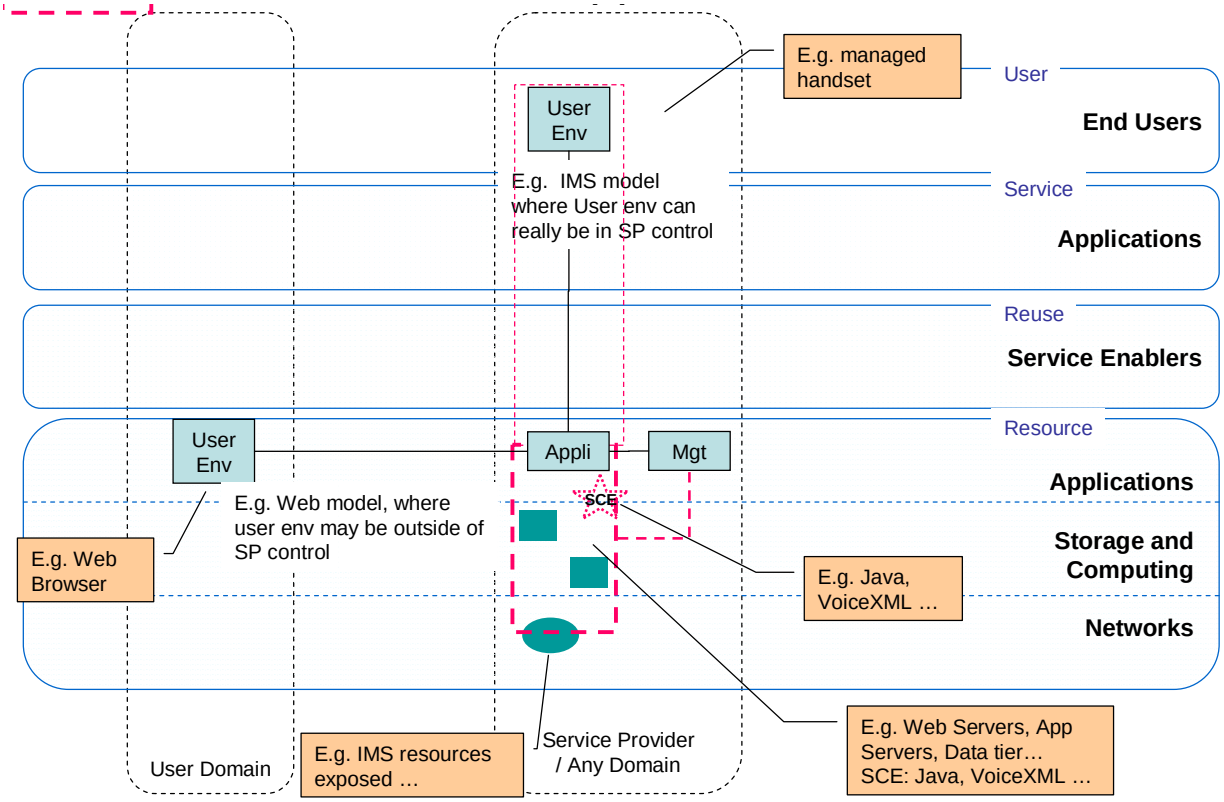


Figure 4-4: Service Delivery in a Single Domain

This use case illustrates a typical deployment model for services currently in place in communication service providers, in which a SON approach is not being adopted. This outlines how a service delivery platform could be used to deploy legacy applications, in which the application does not make use of service enablers. The SP domain owns and hosts resources and applications and is responsible for service creation. The service may contain some

customization capabilities (e.g. ability for the user to provide preferences), but are fully under control of the SP domain.

4.1.3 Service Applications and Service Enablers in the SDP Context

One could blur the functional distinction between an SDP used to deploy service enablers and one used to deploy service applications, just as one could blur the distinction between SEs and SAs themselves. Both instances have much in common in the way of service software creation/generation, deployment into a hosting/execution environment and binding with underlying resources. The same sorts of software development tools could be used for both, and both may re-use pre-existing SEs in the composition of new SAs or SEs, respectively.

The distinctions between the two lie in the following areas:

1. An SDP used to create and deploy service applications:
    - Would involve defining/creating a 'service' wrap for end-user services, including presentation/appearance to end-users, charging/billing options, customization options, subscriber management, user profiles, SLAs
    - May address other commercial aspects of service support relevant for end-user services; e.g. trouble ticketing or call centers
    - May be tied to service syndication and a service marketplace.
    - Would allow for top-of-the-food-chain SAs that can be arbitrarily complicated and whose interfaces/behavior cannot necessarily be precisely or concisely articulated (because for example they involve human resources such as operators dealing with unusual customer situations)
    - May typically involve impacting certain kinds of network (or perhaps even operations) resources that deployment of a SE would generally not. For example, many end-user services involve establishing media sessions that relate to elements such as Session Border Controllers, Media Servers, Electronic Number Mapping (ENUM) databases, Call Session Controllers, etc.
    - Would typically allow 3rd parties to create their own services for end-users.

2. An SDP used to create and deploy service enablers would typically 'publish' a SE as a Web Service so that any potential 'consumers' can be aware of its existence (not typically done for service applications).

While logically we have separated these two concepts, vendors may, and in fact some already do, offer single products that embody their combined functionality. The same product may include additional functionality as well, such as a service execution environment, an operations environment, or even various network functional elements according to some particular architecture such as IMS.

The use case shown in Figure 4-5 brings the ability to expose a Service Enabler in a way that allows it to be composed into a Service Application. The service application may be contained within a single domain or may be part of a mash-up with a Web2.0 application inside the User

domain. The SDP implements an environment where single domain resources can be exposed into a reusable SE, allowing interactions with 3rd-party domains, exposing a functional and non-functional interface towards the service applications, as shown in Figure 4-5.

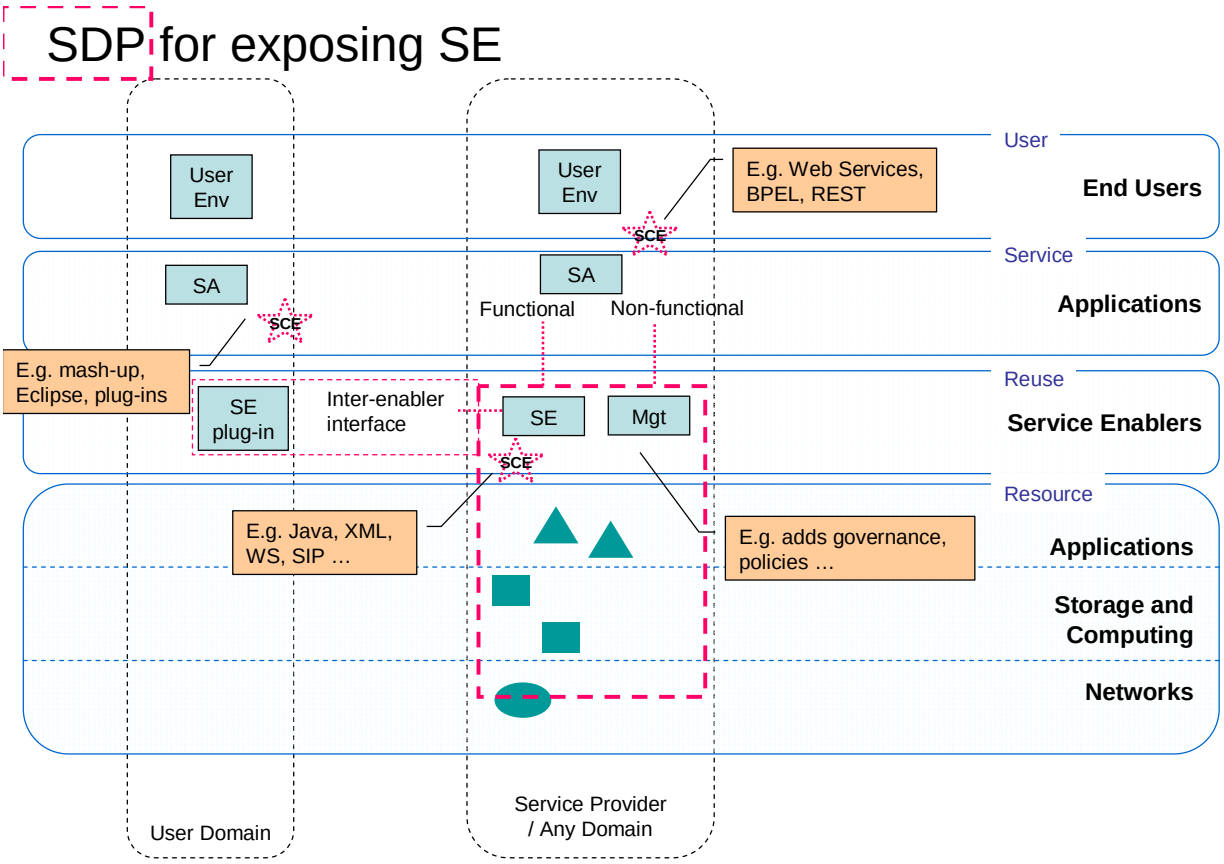


Figure 4-5: Service Delivery and Service Enabler Exposure

Creating and managing a service application provided in this way will require getting access to the SE from a functional standpoint but also to other non-functional properties of the SA (see below).

The SDP in this case brings additional capabilities for the exposure and composition of underlying resources and applications. Re-using an SE may require some components to be deployed by the user domain, like a plug-in through which some functions of the SE can be invoked (e.g. from the User domain you may have access to a REST Application Programming Interface (API) of the SE as well as to an Real Time Transport Protocol (RTP) source of the SE).

Service Creation may occur at different places:

- Creation of the SE (the SCE may have ability to characterize SON-compliant SEs)

- Creation of an Application inside the same domain as the SE, which may use native interfaces. Note this SA may also rely on components inside the controlled User environment
- Creation of an Application from outside of the SE domain, which may need to go through specific capabilities (e.g. security).

4.1.4 Service Delivery Across Multiple Domains

This use case, illustrated in Figure 4-6, is a more extensive view of the way in which a service provider could develop and deploy SON, using an SDP. Given that this concerns multiple providers in the value chain, there are business model aspects to be considered as part of the deployment and operation.

In this case the SDP is used by a 3rd-party application provider domain to compose enablers from different domains into a SON manageable / hosted application. The SDP is an environment where multi-domain SEs can be composed into an application. Although such an SDP could be a 3rd party extension of one of the underlying SE's SDP, it may be owned by a completely different 3rd-party.

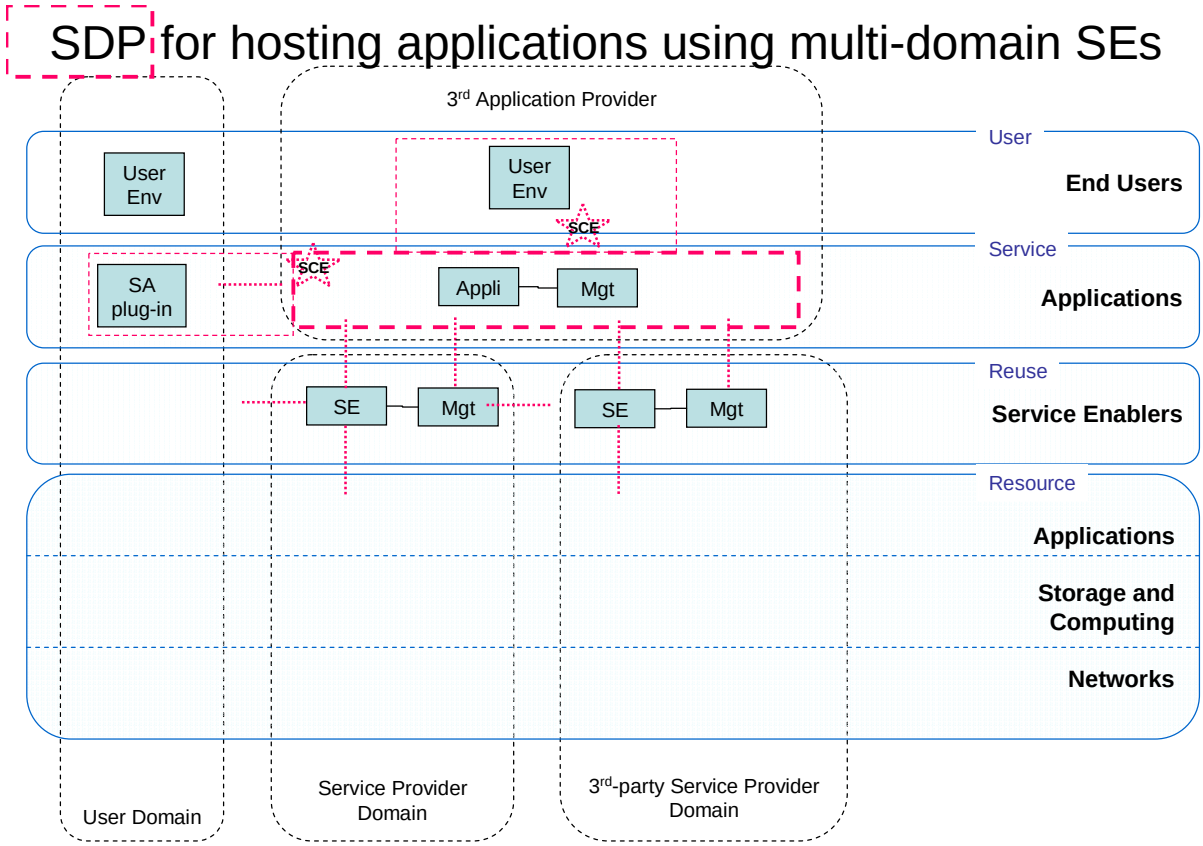


Figure 4-6: Multiple Domain Service Delivery

Service creation in this example is focused on the application experience (and deployment) spanning user domain and application provider domain. The service application may provide service creation as a customization feature on top of a composition enabler.

4.1.5 Service Delivery and the User Environment

The above use cases try to illustrate some considerations about the user domain and where a Web 2.0 approach and style of service creation and assembly (e.g. fast, content-aware, community-enabled …) should apply.

A composed service application can be engineering to allow further tailoring by the user. This can stretch from a simple mash-up by the user which integrates the functionality of the service application with a 3rd-paty application (say from the Web), though to providing the user with a more extensive service creation environment in which the user composes the application from the functionality of service provider's service enablers. Clearly, in an example in which the user is undertaking more extension service creation, the user environment will need to contain the means to undertake software development and integration.

This can be illustrated with the following scenario: a user is willing to set up a service for sharing vacation experience with friends' community. He/She will define which enablers are needed (e.g. video sharing, blogging / logging of activity, friends conferencing, location tracking …), and be able to undertake the service composition by picking enablers from a menu. He/she can also describe the charging preferences (policies) and more preferences like ad-free.

This puts stringent requirements on the 3rd-party Application provider to provide this service. The provider must have extensive knowledge of the contexts in which targeted users might wish to compose services and of the types of enablers required (including discovery and brokering of features based on price). The assembled service must work in a controlled manner and be able to be tested before deployment to the network.

Moreover, an SE may have to expose information to the User Environment outside of the SE's domain control by, for example, deploying plug-ins or direct control interfaces as shown in the use cases above. This means an SDP will have to cope with diverse components and domains and hence could require interoperable standards to deal with this.

4.1.6 Non-functional Aspects of Service Delivery

The above use cases illustrate the needs for service enablers to be engineered to take account of both functional and non-functional requirements. A challenge for service providers is that they

may not know the precise details of the way in which a service enabler may be used by a 3rd-party, for example in a cross-domain composition, including a user application from the Web.

An important objective for SON Service Applications is to enable delivery of such services across domains in a manner where quality of service and experience can be managed accordingly, enabling many business models. Any application doing this must have to reconcile capabilities over different providers.

Non-functional aspects of service enablers are a key subject for standardization in order to allow better cross-domain composability. Examples include: availability, reliability, security (including privacy constraints), capacity, response time, chargeability / billability, composability (policies, contributions to Service Level Agreement (SLA)), user profile dependencies / relationships, customizability (scripting, configuration, etc.), supportability, lifecycle model, system and resource dependency model, and testability.

4.1.7 Standardization Assessment for Service Delivery

Although service oriented architectures provide a set of principles for the way in which organizations can engender re-use across their portfolio, there is also a strong appetite for increasing guidance as to how service providers can create, deploy, operate and maintain their SOA-based services.

The term service delivery platform became used as short-hand for a homogeneous environment in which service providers can offer services based around agile service creation in which service applications are built from re-usable service enablers. Although a look at vendor SDP offerings suggest that there are a number of common elements, the industry has not come close to a standard definition. The TMF's service delivery framework program is concentrating its efforts on the management interfaces (see section 8.4.7) to SDF services. This means that there is a large gap in the SON standards landscape for service providers wishing to procure a standard service delivery environment. Moreover, we see no attempts to standardize the way in which services can be developed and deployed across multiple provider domains. If service providers are to capitalize on the use of services outside of their own domain (in particular services offered over the Web), then further impetus must be given to this area of standardization.

Other areas of standardization will need to occur in order to support the kinds of service creation and delivery outlined in the use cases above. These include:

- User environment, including the use of plug-ins (lifecycle of the plug-ins and how to enable creation of interoperable plug-ins). Plug-ins can apply to service creation, service execution, and service deployment.
- Handling of non-functional properties in single or multi domain composition of enablers

- Interoperable discovery of service enablers (including both functional / non functional aspects)
- SLA / Policy implication of cross-domain combination
- Security / liability aspects

In addition, where an SDP is used as an environment to create and deploy service enablers, the following must be considered:

- SE deployment by a SDP. How SEs are published in a registry, including how they are rigorously described in the registry. How their use may be limited to a certain set of 'consumers' and not necessarily the entire universe of potential consumers. How SEs are gracefully retired. Versioning/update management. How operations resources are made aware of SEs that they need to manage.

- The description of a SE should include a variety of characteristics that may be important to a potential user of the SE (including SA 'users' of the SE). For example:
  - Capacity
  - Reliability/Availability
  - Response time
  - Other QoS-related aspects
  - Interface security options
  - Charging policies for usage of the SE
  - Failure/failover/degradation modes of the SE
  - Level/degree of testing and/or certification
  - What entity created the SE
  - Potentially whether this SE instance is for general use or a beta/testing version
- Given that a primary tenet of the SE concept is shared use, whether there should be mechanisms and rules for allocation of usage across multiple users, e.g., for some guaranteed share of the SE resource capacity for a particular user.

4.2 Agile Service Creation

Agile Service Creation (ASC) refers to the process whereby new products and services are created using methods applied from the software industry that reduce the time from concept to market. The adoption of ASC is aided through the use of SOA tools, technologies and techniques (see Section 3.2), in which all components that are to be integrated as part of the service creation are made available as SOA services. Where these SOA components exist and can be used, the service creation process becomes one of integration, rather than bespoke development. By its very nature, integration of pre-existing component parts is typically a more agile method than one in which specific new component developments are needed as part of the creation process. The commonly-used analogy is with the auto industry in which new models are assembled from component parts that are common to other models and which are integrated with a process built around re-use on an assembly line.

The goals for CSPs in seeking to replicate this method are therefore:

a) The creation of a set of re-usable service component parts, developed and exposed as service enablers. For communication service providers (CSPs), this means applying this goal to the integration of all types of technologies (not just software), including those in use in the network.
b) The creation of an automated method for service enabler integration.
c) The inclusion of the service wrap in the created product or service, using a) and b) above.

Within the overall domain of agile service creation there are 3 main categories:

1. in which CSPs undertake the service creation themselves, assembling components parts (service enablers) that they have created and/or those created by 3rd-parties,
2. in which 3rd-parties undertake service creation using service enablers that they have created and/or those offered by CSPs, and
3. in which users themselves create new services from component parts offered by CSPs and/or 3rd-parties.

Although re-use of the service components and of the service creation process are key to agile service creation, these three types of ASC have different characteristics and requirements, as outlined below.

A common thread which runs through all three types of ASC is that of an open, competitive environment for both services and their component parts – again analogous to the auto industry. Auto manufacturers may create component parts for their own models and also for their partners (and even competitors) as an additional source of revenue. In addition, the auto parts industry seeks to be as manufacturer/model agnostic as possible. The ultimate purpose for CSPs adopting strategies, architectures, and methods for ASC in this way is that the whole market for services grows, in the same way that the market for independent software vendors has grown over the last twenty of so years with the advent of the personal computer. The more computers that are sold; the greater the market for new applications. The more applications available; the greater the value in owning and using a computer, and so the cycle repeats.

Note: For all types of ASC, a number of measures will be applied to check whether the agile methods and the resulting services are fit-for-purpose. Two measures that are in common use across many types of production are a) Right First Time – in which the finished product is tested against the customer requirements with the aim to meet as close to 100% of the requirements as possible, and b) Reduced Cycle Time – in which enhancements to existing products or even the creation of new ones is brought about as quickly as possible. In some cases, these measures can be conflicting. For example, in order to get products to market as quickly as possible some companies adopt an approach where minor defects in the product are

acceptable (one of the Web2.0 principles is that of perpetual Beta in which the product is constantly and rapidly undergoing changes and is never considered finished – see section 3.1.3). Conversely, in order to ensure maximum compliance with customer requirements, it may be necessary to undergo many iterations around the development cycle, increasing the overall cycle time. CSPs wishing to adopt agile service creation and use such measures in the process must balance these conflicting requirements.

4.2.1 Agile Service Creation by Communication Service Providers

CSPs are increasingly turning to agile methods for product and service development, aiming to reduce the time-to-market from years to weeks/months. A number of agile methods concerned with the overall product development lifecycle are in force across the industry in which users and developers participate in methods designed to bring the process of requirements/development/testing towards one of rapid iteration.

In considering the application of technology to this process, one area in which a number of CSPs have concentrated is the part of the process concerning the software integration itself. Where this integration follows a SOA model, that process becomes one of 'assembling' the finished product from a set of ready-made service enablers. As a result, this process is sometimes referred to as Agile Service Assembly (ASA) and has the following underling principles:

- It must be possible to know early in the lifecycle if a product would not be profitable and apply any lessons to the next iteration. This enables the 'fail fast' approach to product development that lowers sunken cost in products that are either not profitable or not technically feasible.
- The process of service assembly must re-use technology and operational investment and eliminate the creation of product stovepipes. As far as possible, new products must re-use the existing portfolio, preferring re-configuration and assembly to the use of new engineering.
- The process must encourage innovation and allow the rapid introduction of new technologies.
- Service assembly must allow product designers to be much more closely involved in the whole process including technical development. In order to prevent an increase in staff costs, the whole process must be de-skilled, without losing ability to innovate.
- The components for service assembly must encourage partnerships and enable the use of third –party services through open architectures and the use of standard tools and techniques.

4.2.1.1 **The Agile Service Assembly Process**

The Agile Service Assembly Process (ASAP) provides a phased approach to take the product designer from idea through to launch. The 5 phases in the ASAP are:

- **Conceptual Modelling** – enables the exploration of the possibilities for the service provided by the available capabilities and services, and enables any gaps in functionality required to meet the minimum marketable feature set of the service to be identified.
- **Concept Prototype** – achieves an early executable prototype that will prove the basic design and enable the market and product managers to check whether the solution is what they envisaged.
- **Concept Trial** – provides the means to build and run a trial of the service with representative customers to assess market attractiveness and to gain valuable information about the way in which the service will perform once launched.
- **Implement** – scale, develop and integrate the service to support expected volumes to a hardened, fully functional service that meets non-functional requirements and is ready for user acceptance and operational readiness testing (UAT and ORT).
- **Test, Trial and Launch** – undertake UAT/ORT, deploy and go live.

Each process phase may include iterations. One of the explicit intentions of the ASAP is to enable a product to be developed to any one of these stages and then dropped if further development is not warranted. Reasons for stopping development can include an unprofitable business case, technical feasibility, and lack of market interest, among others.

Key aspects of the process involve rapid time-to-market and lower product development costs. Figure 4-7 below shows the way in which the ASAP affects the proportion of budget required to develop the product as a function of time.

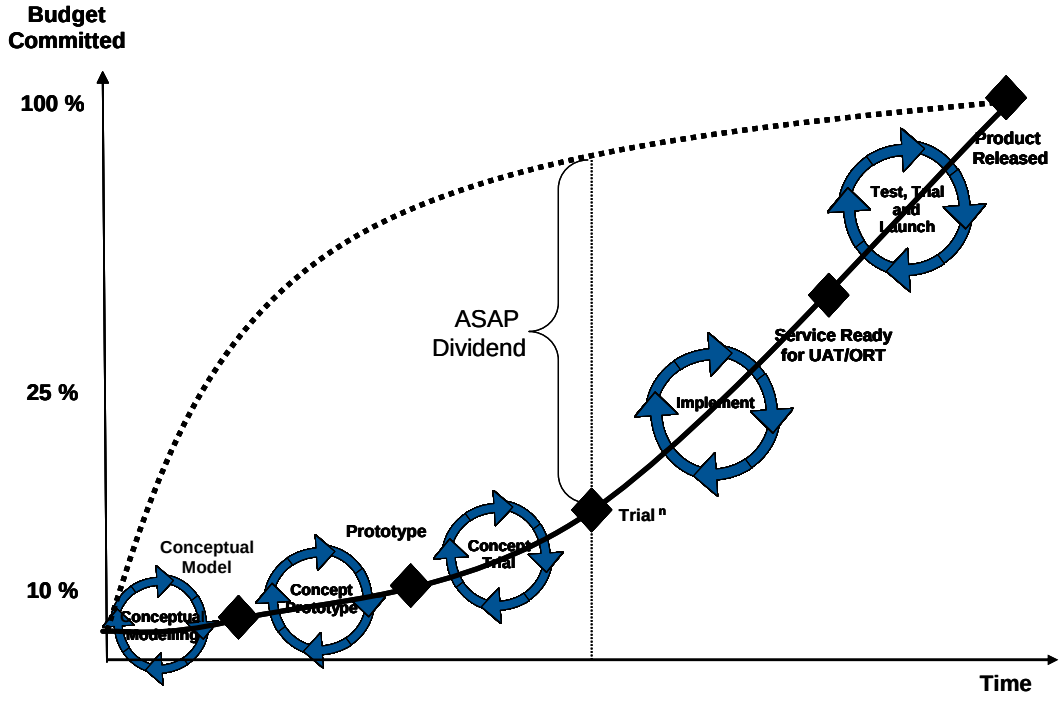


Figure 4-7: Agile Service Assembly Process

The key feature of this diagram is the shape of the curve illustrated with the solid line. It shows that a major proportion of the product development activity can be accomplished with a relatively small outlay of budget expenditure. ASAP allows a product designer to get as far as service trialing with perhaps less than 20% of budget used. This compares extremely favorably with the situation of traditional product development, shown with the dotted curve, in which a major proportion of a product's budget may need to be used by the time the product is ready for service trial. The 'ASAP dividend' available using this approach may be as much as 50% of the product development budget.

The whole process is expected to be complete within weeks/months (depending on product complexity) rather than the months/years that is common for traditional product development. This allows much greater flexibility in product development and allows a higher degree of more speculative or risky ventures to be undertaken. If a new product idea gets as far as service trial and is then deemed not to be worthy of taking to full service launch, then a) the lessons learned from the modeling, prototyping and trialing can be applied to the next new idea; b) the relative investment is low; and c) the project has taken a relatively short time to provide the information required to start anew, that is, it is displaying one of the characteristics of service innovation on the Web, it has 'failed fast'.

4.2.1.2 **The Use of Business Process Modelling in ASAP**

The use of SOA tools and techniques for service assembly comes from the world of software engineering. In service-oriented networks, even when we are constructing new products that include network functionality, the way in which that functionality is built into the product is by treating it as a software function. Constructing new products is therefore about software engineering, for which we use object-oriented modeling. Object-oriented modeling has been at the bedrock of software engineering for over fifteen years and is entirely compatible with service-oriented architectures. In order to construct the service application, however, there is also the need to develop the business process that is enacted by the service. This might typically include how the user logs in, how the service is used, how the service deals with any service failures, etc. Object-oriented modeling is less suited to the construction of a business process, and for this we need to include a complementary approach known as business process modeling.

Business process modeling has grown from the world of business process automation, in particular the interaction of business activities across business-to-business interfaces. The techniques involved have roots in workflow management solutions, but have more recently been applied to the area of orchestration.

Orchestration is a term that has come from the world of Web Services to refer to the way in which two or more Web Services can be strung together in the form of a workflow or

business process. Figure 4-8 below provides a graphical representation of an orchestrated business process.

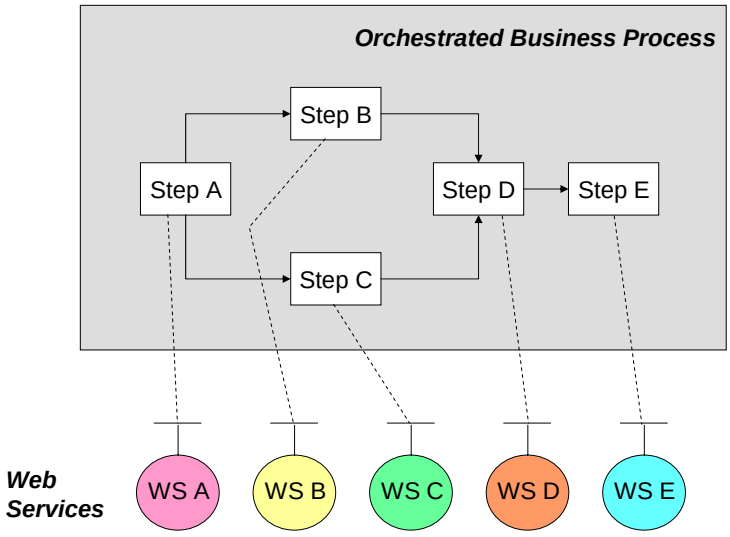


Figure 4-8: Orchestrated Business Process

The process itself consists of a series of five steps (A to E). The process begins with step A and ends with step E. The process steps themselves are each enacted by invoking a Web Service. In this example Step A is enacted by invoking Web Service A, step B is enacted by invoking Web Service B, etc. The two key elements needed to describe an orchestration are:

1. The way in which the sequence of steps operates, including all forms of conditionality and exception handling.
2. The way in which the steps are enacted by the Web Services.

Separation of these two constructs allows the process itself to be described independently from the way the process is enacted. This separation is a key example of the concepts of modularity and re-use in a service-oriented architecture.

Along with the basic standards for Web Services (XML, SOAP, and WSDL) a specification has emerged for Web Services orchestration – know as Web Services business process execution language (BPEL for short). The BPEL specification supports the two key elements of orchestration described above, but in addition, it provides for the use of BPEL as a *model* for describing exchanges that characterize business partnership interactions. It does this by:

- Using standard Web Services to invoke the partner services
- Exposing the resulting business process as a Web Service

- Creating a language that is portable onto any platform supporting the specification.

BPEL is based upon XML – in common with other Web Services specifications - and it invokes the Web Services that enact the process steps using the specifications for SOAP and WSDL. XML has the benefit of being machine-readable (therefore able to be used as a standard messaging notation) and is human-readable (aiding with software construction, debugging, and VV&T).

The basic building blocks in BPEL are not dissimilar to other languages that can be used to describe a series of activities or a workflow. However, the fact that BPEL is built upon the basic Web Services specifications, makes it the ideal choice for Web Services orchestration. Moreover, because BPEL allows the process *itself* to be expressed as a Web Service, it is possible to build hierarchies of orchestration, e.g. where a service orchestration consists of a process involving service building blocks, which themselves are based on orchestrations.

Many service assembly toolsets that employ BPEL for service orchestration also provide a more graphical representation of business process modeling, based upon the specification for business process modeling notation (BPMN). This allows an easy construction of the process which may then be rendered into BPEL for subsequent refinement and testing. Furthermore, as BPMN becomes more established as the standard for process modeling, we expect to see rendering directly from BPMN into the native code that will run the application in deployment.

4.2.1.3 **Service Assembly Toolset**

Adapting the concepts and technologies described above to the goal of service assembly provides us with the principles against which we would choose an automated toolset to allow CSPs to rapidly create new products. Such as toolset will provide the following:

- Investment efficiency – the toolset will provide the means for and will encourage rapid iterative development, allowing product mangers and service assemblers to get to early concept trials for new ideas.
- A collaborative development environment – providing one process and toolset linking product designers, service assemblers, platform developers, operational managers, VV&T personnel, etc.
- Model-driven environment – that enables rapid concept validation and service development with as little investment commitment as possible. This is particularly important where the concept turns out not to have a downstream route into a launched product.
- Extensive re-use of sunken investment – ideally, every new product that can't be developed through data re-configuration can be developed through the assembly of components that have already been deployed.

The essential components of the toolset are shown pictorially below and include the following items:

- A visual design tool supporting BPMN, with a behavior simulation capability included;
- A BPEL design tool that produces (e.g. Java 2 Platform Enterprise Edition (J2EE) or .NET) translations of the BPEL, and that transparently consumes output of the BPMN tool;
- A Web Services repository and catalogue that maintains service WSDLs, component interfaces, and links to live services;
- An integration of these components in a single Integrated Development Environment (IDE).

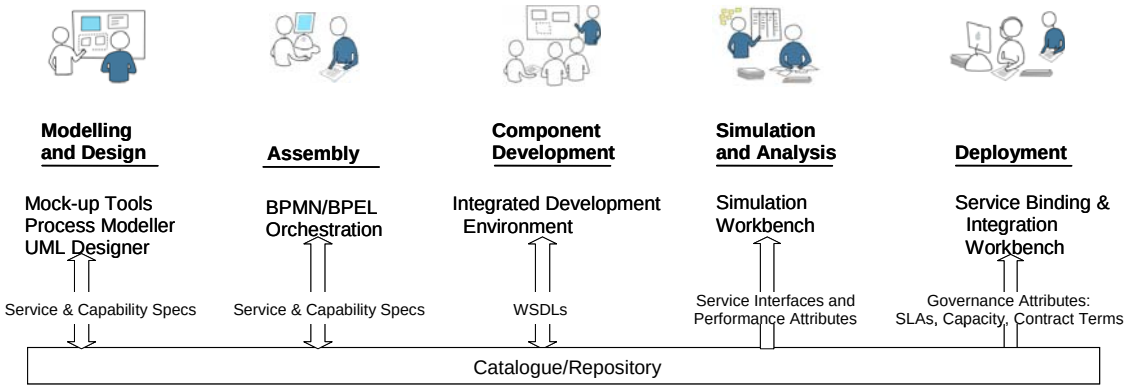


Figure 4-9: Service Assembly Tool-set

The way in which these components are used is as follows:

- BPMN Tool – The BPMN tool is used by a service assembler working with a product designer to design the process flow of a new service. Existing components are available in the tool (with their associated behaviors) for this purpose. Newly defined components that don't yet exist can be defined and modeled in the tool and incorporated into the service definition. The new service definition can be actively simulated, testing the behavior of the service and the interaction of the features of the service with other services.

- BPEL Development Tool – The BPEL development tool enables the elaboration of the service designs created in the BPMN tool into working code, including the orchestration of the service components that are part of the design. The formal expression of the process flows is done in BPEL, however the code created is Java-based, .Net-based or other commonly-used code.
- Services Repository – The Service Repository and its associated catalogue is required as the authoritative source for the publication of new and existing services and components. Services and components published in the Repository can be referenced from the design palettes of the BPMN and BPEL tools.
- Integrated Development Environment – This provides for tight integration amongst the tools to allow rapid iteration of service designs through the lifecycle, while having immediate visibility of available capabilities and interfaces.

4.2.2 Agile Service Creation by 3rd-parties

The use of the service assembly process and toolset described above refers to CSPs assembling their own services from component parts that they and/or their suppliers have made available for integration. However, as outlined above, CSPs are increasingly encouraging 3rd-parties to develop their own products and services using a similar approach, where the CSPs themselves provide some of the building blocks.

A CSP may have developed a service enabler for use internally as a re-usable component in its products and services. An authentication service enabler is a good example that would have wide applicability across a wide range of services. The CSP may be able to make the authentication service available to 3rd-parties and charge for its usage. A model to do this might include hosting authentication as a Web Service and providing a charging mechanism for its usage. 3rd-parties can register for the service, create an account and then integrate authentication into their offerings.

More recently however, CSPs have launched their own software development kits (SDKs) aimed at external developers. The SDK is a set of developer tools, documentation and sample applications that expose a series of service enablers (such as authentication). It is an easy way for a 3rd-party developer to begin to take advantage of read-to-use service enablers provided by a CSP, without the developer being hidebound by the CSP's own development processes or toolsets. Typically, the SDK integrates into the integrated development environment that the 3rd-party is using and the service enablers appear as components that can be integrated into any service development. Also included is the means for the 3rd-party developer to enter into an SLA with the CSP that describes how the service enablers will operate and any resultant charges.

Revenues can accrue to CSPs via a variety of business models which may include traditional usage-based charges but may also include more innovative revenue generating methods, as determined by the 3rd-party developer. In offering services to users based upon developments using the SDK, the developer is acting as a third-party service provider. They are free to charge

their customers in any way they choose and CSPs may have the option of sharing in those revenues using a method agreed between the two parties.

4.2.3 Agile Service Creation by Users

The type of service assembly described in Section 4.2.1, and even the use of SDKs as described in section X, requires the use of a toolkit that is part of an integrated development environment (IDE). Typical IDEs involve the use of source code editors, compilers, debuggers, etc – that is, the type of tools used by software developers. Even the most advanced and intuitive of IDEs require users to have at least some software engineering skills in order to be effective.

However, the world of Web2.0 really hits home when the ability to assembly innovative new services lies with users themselves, especially when users do not need such skills. Although the early days of Web2.0 required knowledge of such software technologies as asynchronous Java and XML (AJAX), more recently, we have seen the emergence and rapid take-up of user-based service assembly environments which require little knowledge of software programming. A typical example is Yahoo Pipes™ – "a powerful composition tool to aggregate, manipulate, and mashup content from around the web". Through the use of a simple drag-and-drop interface, accessed via a user's browser, new services can be quickly created by assembling functions and data sources together. The user needs no development environment and little in the way of programming experience. The figure below shows a basic example of the use of Yahoo Pipes™.

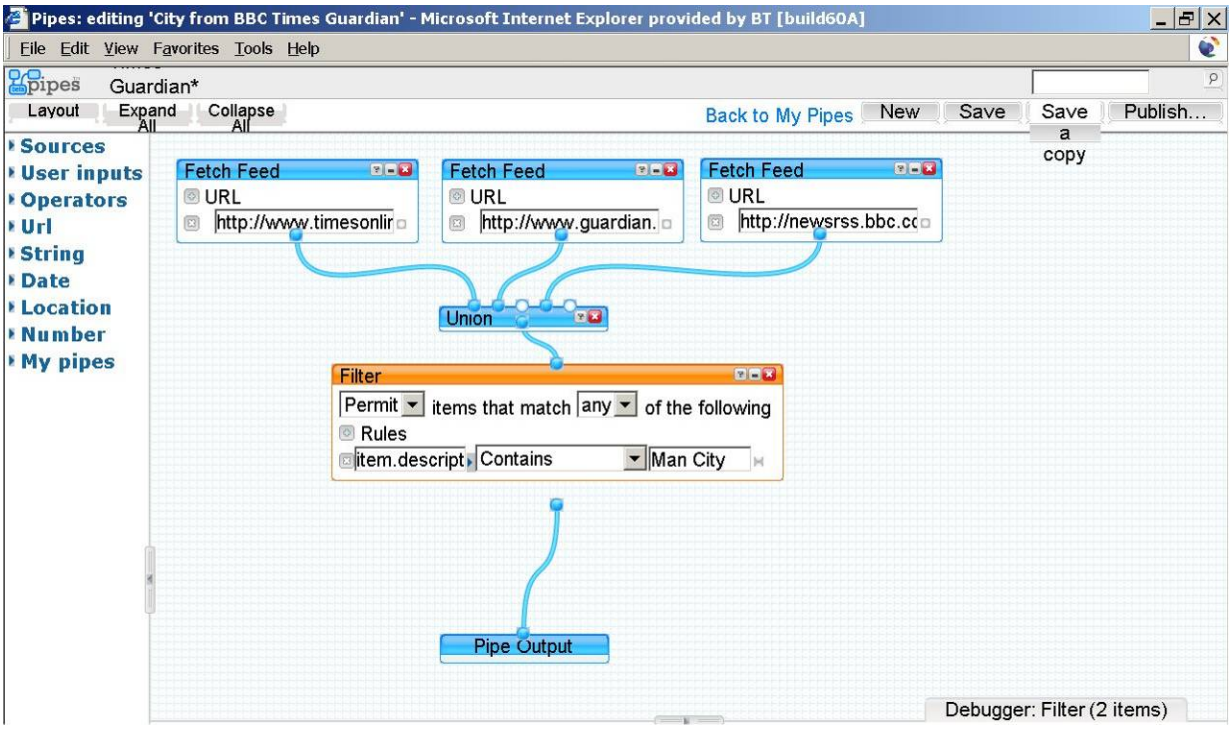


Figure 4-10: Yahoo Pipes™ Example

The pipe works by taking the data from three news feeds (Times, Guardian, and BBC), joining them together (with a Union operator), applying a Filter that allows through only items whose

descriptions contain the term "Man City", and sending the output to the browser. The output will provide a list of news stories about Man City (a UK soccer team) which the user can then click to access the further data. The left hand side of the diagram shows a list of building blocks (Sources, User Inputs, Operators, etc) with which the user can create new services. One can imagine a whole host of data sources that can be integrated together as in the example above, and when combined with more functional building blocks (e.g. the ability to send the data output by text to a mobile phone), it is easy to see that quite innovative service creation is now within the grasp of a non-technical user community.

A number of CSPs are currently in the process of developing and launching environments that allow their users to undertake service creation in this way.

4.2.4 Agile Service Wrap

The service wrap for a CSP's products provide the means for customers to order service, for service problems to be fixed, and for there to be some means of billing. Service wraps are defined in terms of business processes, and can be characterized into one of:

1. Concept-to-market – describing the process for taking an initial idea for a new product right through to its launch (and eventual withdrawal) into the marketplace.

2. Lead-to-cash – describing the process for taking customer leads and bids, turning them into firm orders; taking, validated and fulfilling the order; and putting in place the means to bill for service.

3. Trouble-to-resolve – describing the process for accepting, diagnosing and fixing problems to do with service.

In an ideal world, all of a CSP's products would use identical processes, but the reality is that some changes to the way in which a process works is inevitable given the wide diversity of the product portfolio. For example, consider two steps in the lead-to-cash process which may be enacted by process steps 'capture order' and 'validate order'. For a simple consumer product the customer may order on-line and an automated system validates the order data. For a complex business outsource solution, the customer may require a number of visits from product managers and the order itself may need extensive validation by an order management team.

Cost can be taken out of a CSP's operations by automating the business processes. In the same way that the time taken to develop new products using a traditional method can stretch into years, so too the time to develop the service wrap for such products can also be very lengthy, especially when the process is automated via the use of OSS. The traditional method of developing OSS to support a new product is shown below. In this method, the description of the new product is manually documented and then assessed across a series of OSS programs to determine where OSS developments need to be made. One or more OSS platforms may be affected and within each platform, one or more systems may need development. OSS development activity then results, and may be undertaken by developers, systems integrators, or by the systems vendors themselves. The process from product specification through to

defining the development requirements is very manual in nature and as a result can be time-consuming, costly, and error-prone. This method of development is unsustainable if the goal is agile service creation.

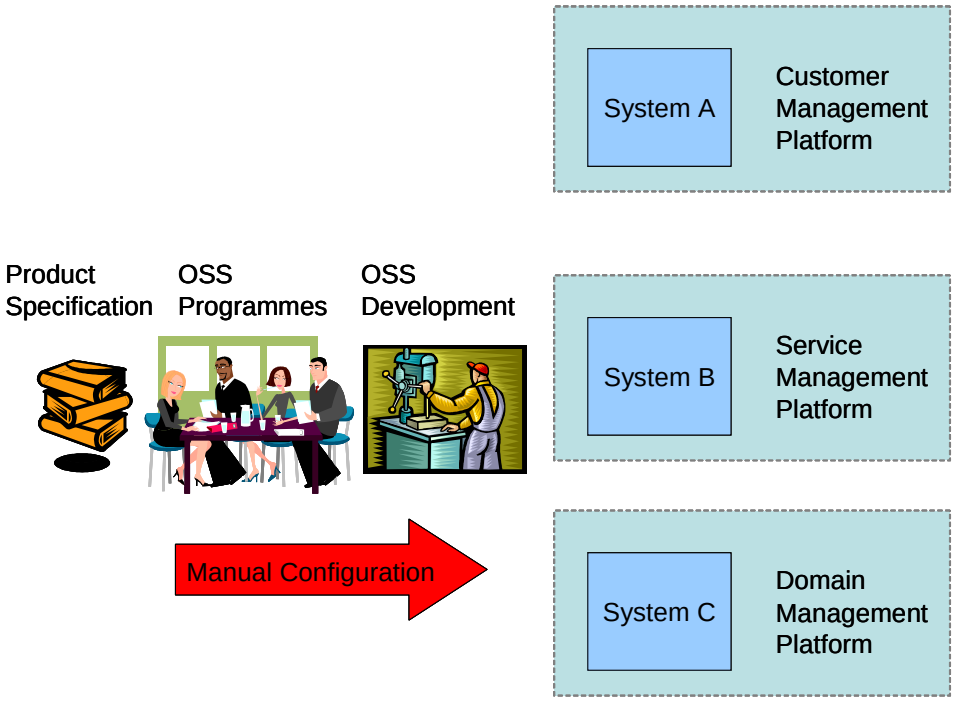


Figure 4-11: Traditional OSS Development Method

The challenge for CSPs is to ensure that the service wrap for any product can be created in the same timeframe as that for the product itself. Clearly, therefore, as we seek to reduce product development times down to weeks/months, we must do the same with the service wrap. This means that the time-consuming parts of the process shown above must be replaced with a quick and automated configuration process.

Many off-the-shelf support systems (particularly those procured more recently) have configuration interfaces that allow a data-driven method of effecting change in the way the system behaves. These interfaces can accept a data file (written for example in XML) that can be used by the system to alter data structures that determine system behavior. For example, consider a product that is ordered by calling into a call centre. In the call centre, the customer service agents will use systems that are part of a customer management platform to capture and validate the order. The system will present the agent with a series of screens into which the agent will enter the customer and order details. If the CSP creates a new product it is likely that it will be ordered in a slightly different way using slightly different data. The order entry screens are also likely to differ. However, the basic process of order entry is the same, as will be most of the data about the customer (name, address, etc). The change to the system therefore, to accommodate the new product, will be to change the screens so that the new order data can be entered. In software engineering terms this is a relatively straightforward change, and can be made by providing an XML file into the configuration interface of the system. The process for system development becomes highly automated and able to be accomplished quickly.

In order for the data required by the configuration interface to be the right data for each new product, that data must be generated as part of the agile service creation process. In our example of order entry, the service creation process must provide the data describing the way in which the order entry screen will be used to help capture details about the order. This data is therefore a vital part of the product definition. The figure below shows how the traditional OSS development method can be replaced by an automated configuration method based upon data generated as part of the service creation process and provided as part of the product definition. The diagram shows the use of configuration interfaces on each of the systems used to automate the service wrap. The manual aspects of the development process, based upon the product specification, are removed.

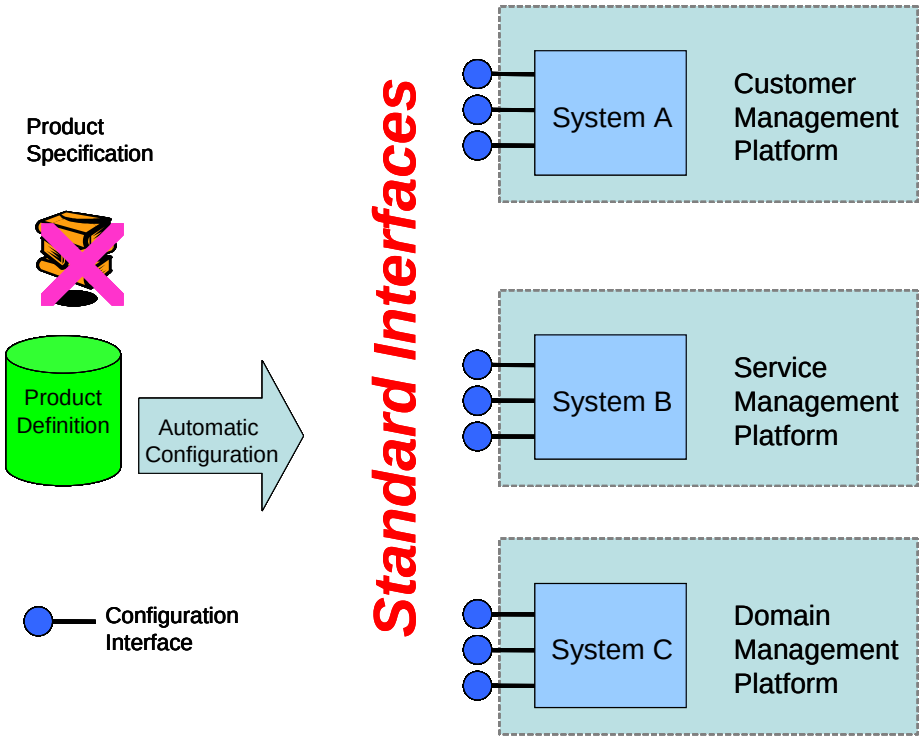


Figure 4-12: Automated OSS Configuration

The OSS estates of most CSPs have been developed and procured over a number of years, and a number of vendors are involved in the provision of systems. In order to prevent the use of multiple types of proprietary configuration interfaces, a set of standards should be developed to support this form of service wrap development.

4.2.5 Standards Assessment of Agile Service Creation

Service oriented architectures deliver the goal of agile service creation by allowing services to be created from easily integrated component parts. These component parts – service enablers – can be deployed with open, standard interfaces that allow the integration to be platform and language independent. The most popular standards for service enabler interfaces are based around the Web Services technologies. Since their introduction in the early part of this century,

there has been widespread adoption of Web Services through the industry and they provide the fundamental technologies required for deploying service oriented networks.

However, the basic Web Services technologies provide only part of the solution. Service providers wishing to deploy SONs must chose between further extensions to the basic specifications or the adoption of more controlled environments such as software development kits. These options are explored below.

4.2.5.1 **Web Services**

Web Services is a phrase used to describe the way in which services can be exposed and used on a network, built around the use of technologies such as extensible mark-up language (XML). Web Services concern the way in which software communicates. Software can come in many forms from a simple script on a personal computer, to an application on a networked server, through to a large operational support system running on a mainframe computer. We can view these scripts, applications and software systems as software components. Consider the following characteristics of such a software component:

- It can DESCRIBE itself - so that other components can understand the functionality it offers and how to access that functionality.
- It can allow other components to LOCATE it - so it can be used when required.
- It can be readily INVOKED whenever another component wishes to use its functions.

If such a software component were installed in a network, its services could be used by other software components. Since 2000, a number of technologies have matured and become world-wide standards to allow software components to be deployed in this way. These components are said to be providing a 'Web Service'. An example is a Web Service that returns a credit rating when provided with a user's ID, or an order handling system that provides user and usage data to a billing Web Service which then returns the user's bill.

Web Services can also be integrated together to provide greater value-add. For example, a travel management Web Service may make use of the capabilities of Web Services providing car rental, hotel bookings and flight reservations. Or a video-on-demand Web Service may make use of a communication service that delivers the video stream (which may present itself as a Web Service).

4.2.5.1.1 Web Services Technologies

A Web Service is one that can DESCRIBE itself, be LOCATED, and be INVOKED. The Web Service description is provided by WSDL, it can be located by using UDDI and its functionality invoked using SOAP. These three technologies are fundamentally built upon the common data description standard XML, all of which are described below.

Extensible Mark-up Language

XML is a standard specification for defining the content of a computer message. If a software application writes its output in XML and another application is capable of interpreting XML, then it can read the output and act on it. The way XML does this is via the use of tags. To illustrate this, if you go to the View menu of an Internet browser and select Source (in Microsoft Internet Explorer), you will see the source HTML of the web page you are reading. The content of the web page will be displayed within a number of tags. Each tag will have a name, enclosed within opening and closing angled brackets, e.g. <head>. The tags 'mark up' the content - hence why HTML and XML are called mark-up languages. The following is an example of content, marked-up using the specification for XML:

```
<customer>
        <name>Fred Smith</name>
        <address country="UK">
                <street>93 Example Avenue</street>
                <city>Ipswich</city>
                <region>Suffolk</region>
                <postcode>IP50 9BT</postcode>
        </address>
        <status purchases=”2” last-purchase=”12-05-99”/>
</customer>
```

Figure 4-13: XML Marked-up Content

This is displaying the content for a customer, as seen by the opening tag <customer> and the closing tag </customer>. The other tags are similarly easy to understand, which gives XML one of its main attributes - XML content can easily be made human-readable. More importantly, however, because the content is defined within the tags, which have a strictly defined structure, they are also machine-readable.

The importance of XML to the computing world can be compared to the importance of natural language to humans. If one wishes to convey meaning to another person in a message given to them, and one wishes them to correctly interpret the intended meaning (and perhaps act on it), then a commonly understood language must be used. The language used must have form (spoken, written, etc), structure (words, sentences, etc), syntax (the grammatical arrangement), and semantics (meanings and connotations). Two computers communicating have the same requirement.

If the two communicating entities use a different language, the only way they can communicate is via an intermediary that does a translation between the two languages. In a complex computer system, where there are a number of communicating entities, the use of a common language can drastically reduce the cost of development and maintenance as well as improving its performance.

The software industry has spent countless millions developing software that undertakes translations between different languages. Many attempts have been made to develop and encourage the use of common languages between computer

applications, most of which have failed. However, the success of the world-wide web, and the use of HTML as the common language, has been one of the most successful. It is the success of HTML which has encouraged the use of mark-up languages in general and which has led to the rapid uptake and success of XML.

Simple Object Access Protocol

With XML we have a common way to represent the content of a message sent between one software component and another. SOAP builds on this by providing the means by which one software component can invoke the functionality of another, using message passing between the two as the means of invocation.

The SOAP protocol includes the following:

- An envelope that defines a framework for describing what is in the message and how to process it.
- A set of encoding rules for expressing instances of data types within the message.
- A convention for representing the way in which the procedures (or methods) of the software component can be called, and their responses.
- A binding convention for exchanging the message using a communications protocol.

Version 1.1 of the specification for SOAP was issued by the W3C in May 2000 and includes the binding convention for the HTTP. Although SOAP messages can be transferred using other protocols, the fact that HTTP was chosen reflects the intention to make the protocol available over the Internet.

SOAP uses a request-response mechanism in which one software component makes a request to another software component which then provides a response. Both request and response are transported in the form of XML documents.

SOAP provides a simple but powerful means for one software component to invoke action on another via the use of message interactions. The specification for SOAP is a world-wide standard, administered by the W3C, currently at version 1.2. All software vendors have agreed to implement this specification, which means that a software component installed on one type of platform can communicate using this method with another component on any other type of platform. Note: references to SOAP no longer describe it as being an abbreviation for simple object access protocol – it is now simply SOAP!

Web Services Description Language

XML and SOAP provide the means for one software component to invoke the functionality of another over a network. In order to undertake this integration, the following are also needed:

- Information on all available functions, including their calling parameters.
- Data type information for all XML messages, including the value specifications.
- Binding information about the specific transport protocol to be used.

- Address information for locating the specified service.

A software developer undertaking an integration of software components might use the documentation for each component to get this information. Extracting this information for a complex integration would be time-consuming and prone to human error, thus increasing the cost of integration. Ideally, this could be undertaken automatically by integration software. WSDL has been developed for this purpose. A WSDL file is an XML document that provides the information listed above about a software component. Using WSDL, a software component can invoke any of the available functions of a Web Service. With WSDL-aware tools, this process can be entirely automated, enabling applications to easily integrate new services with little or no manual code.

Universal Description, Discovery and Integration

If one wishes to use a software component and I know the location of the WSDL file, one can simply point my development software to the file and implement the integration. This assumes that one knows the location of the WSDL file and one know about the party responsible for the software. If creating a business-critical application, one will be concerned about the availability, performance, and other non-functional attributes of the service described by the WSDL file. When developing the application, one may wish to assess the merits of a number of software components supplied by different companies. UDDI provides this extension to the basic Web Services technologies by allowing the means to create a registry of Web Services. This takes Web Services into the realm of companies doing business with each other over the Internet.

The UDDI specification enables companies to quickly, easily, and dynamically find and transact with one another. UDDI enables a company to:

- Describe its business and its services.
- Discover other companies that offer services.
- Integrate with these other services.

For example, if one develops a service that relies on a credit checking function in order to validate my customers, a UDDI registry can be used to find that function. The request to the registry would be for a credit checking function and other requirements such as cost limits, security needs, performance criteria, etc. The registry would then propose one or more companies that provide such a function may also provided allowing the choice of a preferred supplier.

The specifications for UDDI allow the creation and use of a registry containing information about businesses and the services they offer. The information is organized as follows:

- *Business Entity*. A business entity represents information about a company. Each business entity contains a unique identifier, the company name, a short description of

the company, some basic contact information, a list of categories and identifiers that describe the company, and a URL pointing to more information about the company.

- *Business Service*. Associated with the business entity is a list of business services offered by the business entity. Each business service entry contains a business description of the service, a list of categories that describe the service, and a list of pointers to references and information related to the service.
- *Specification Pointers*. Associated with each business service entry is a list of binding templates that point to specifications and other technical information about the service. For example, a binding template might point to a URL that supplies information on how to invoke the service. It is also possible to use these pointers to service level agreements that describe the contractual nature of the usage of the service. The specification pointers also associate the service with a service type.
- *Service Types*. A service type is defined by a tModel. Multiple companies can offer the same type of service, as defined by the tModel. A tModel specifies information such as the tModel name, the name of the organization that published the tModel, a list of categories that describe the service type, and pointers to technical specifications for the service type such as interface definitions, message formats, message protocols, and security protocols.

Although the first versions of these specifications have been available since 1999, the use of registries for locating Web Services remains relatively immature.

4.2.5.1.2 The Role of Web Services Standards

The technologies described above owe their creation and existence to the development community in companies across the IT industry. Microsoft and IBM were pioneers in obtaining the basic agreements to work on the same set of specifications. Ultimately, however, the success of Web Services has been due to the efforts of the international standards bodies in turning the specifications into formal standards. Three of these stand out:

World-wide Web Consortium

The World-wide Web Consortium (W3C) was founded in October 1994 to lead the Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has around 500 member organizations from all over the world and has earned international recognition for its contributions to the growth of the Web. W3C is responsible for administering the specifications of many of the basic web services technologies, including that for XML, WSDL and SOAP. See Section 8.1.8 for additional information on W3C standards affecting SON.

Organization for the Advancement of Structured Information Standards

OASIS is a not-for-profit, global consortium that drives the development, convergence and adoption of e-business standards. It has more than 400 corporate and individual members in 100 countries around the world. OASIS and the United Nations jointly sponsor ebXML, a global framework for the use of XML in e-business data exchange. OASIS operates XML.org, a community clearinghouse for XML application schemas,

vocabularies and related documents. See Section 8.1.4 for additional information on OASIS standards affecting SON.

Web Services Interoperability Organization

WS-I is an open industry organization chartered to promote web services interoperability across platforms, operating systems, and programming languages. The organization aims to provide resources for any web services developer to create interoperable web services, and verify that their results are compliant with both industry standards and WS-I recommended guidelines. Unlike W3C and OASIS, WS-I does not actively specify standards - its emphasis is on providing the following:

- Profiles - sets of web services specifications that work together to support specific types of solutions.
- Sample Implementations.
- Implementation Guidelines - recommendations for use of specifications in ways that have been proven to be most interoperable.
- Sniffer - tools to monitor and log interactions with a web service.
- Analyzer - tools that processes sniffer logs and to verify that a web service implementation is free from errors.

4.2.5.1.3 Web Services Extensions

The technologies described above provide the fundamental specifications required to create services that can easily be integrated together. Due to their platform and language independence, these technologies have allowed organizations to take a number of important strides down the road of service oriented architectures. There is a fair degree of interoperability across the industry and Web Services have been far more successful than previous attempts at distributed computing such as Common Object Request Broker Architecture (CORBA).

In many ways, Web Services have revolutionized the development of client-server interactions. Web Services allow any client to create an interface to interact with the server component without explicitly introducing client side code. Using WSDL, most software development environments now support automatic client side proxy generation which, from a software programming perspective, makes a SOAP-based Web service call as easy to call as a local library call. Moreover, Web Services provide the ability to perform a concept known as run-time binding in which the client chooses which web service will be used at run time. UDDI complements the run-time binding features by providing a machine searchable directory standard, with the intention of fostering a globally connected catalogue of services and service providers.

However, for organizations to fully realize the goals of true distributed computing, the basic Web Services technologies are necessary but insufficient. A wide range of additional specifications have been developed that build on the basic technologies and provide additional structures to be put in place for security, routing, addressing, policy management, trust, etc. These specifications are in varying degrees of maturity and are maintained or supported by various standards bodies and entities.

Specifications may complement, overlap, and compete with each other. These extensions to the basic Web Services specifications are occasionally referred to collectively as "WS-*", though there is not a single managed set of specifications that this consistently refers to, nor a recognized owning body across them all. The reference term "WS-*" is more of a general nod to the fact that many specifications are named with "WS-" as their prefix (e.g. WS-Security).

Although not designated with the WS- prefix, BPEL can be considered to be one of the Web Services extension technologies. As outlined in Section 4.2.1.2 ,BPEL provides an XML-based specification that allows Web Services to be orchestrated together in the form of a business process. This is typical of the kind additional specification that are required to form a more structured architecture for Web Services, allowing them to be used as part of the more formal service creation programs in use in communication service providers.

Due to the relative immaturity of the WS-* specifications, there is far less interoperability across the industry than with the fundamental technologies. Moreover, the construction of a robust and scalable Web Services architecture that provides platform and language independence in addition to structures that provide addressing, security, federation, trust, etc., has become far more complex. Also, many of the WS-* extensions require client-side implementations, thus removing one of the fundamental benefits of basic Web Services – that of automatic client-side proxy generation.

As a result, we have seen the emergence of client side software development kits starting to be adopted as a means to manage the complexity and abstract it from client side developers, as outlined in Section 4.2.1.3. Whilst complexity can be managed in this way, some benefits are lost. The run time binding feature and platform independence are compromised. You need to supply an SDK for each platform / language supported, however this is not quite so much an issue since the main developer communities can be addressed with an SDK for Microsoft .Net , PHP, Python and Java. These SDK implementations provide a controlled environment in which clients and servers interact within the confines of the specific deployment. The open aspect of distributed computing is therefore reduced and a number of SDK implementations are questioning the use of technologies that support open communications – that is, the basic Web Services technologies. As a result, a set of more light-weight technologies are being used within such environments, based around an architecture known as REST.

4.2.5.1.4 Representational State Transfer

REST is a term used to describe an architecture style for communicating systems. It provides a collection of principles which outline how resources are defined and addressed. The term is often used in a looser sense to describe any simple interface

which transmits domain-specific data over HTTP without an additional messaging layer such as SOAP. Services using the REST approach are often termed RESTful.

The architectural style embodies three principles that determine the reason that REST has been named as it has. If one uses a web browser to link to a URL (e.g. www.bt.com/resources/broadband) then server will return a ***representation*** of that broadband resource. This places the client on which the browser is running in a certain ***state***. Typically, the server will return a series of web pages in HTML. The browser, parsing this HTML, is then placed into a new state, i.e. it has undergone a state ***transfer***.

This approach is the fundamental way in which the Web works. Browsers traverse URLs on the Web and the data returned by the Web servers alter the state of the client. This concept can be extended not just to data on the Web but also functionality. Developers can use this approach such that accessing a URL kicks off a search for information or makes a phone call.

An important concept in REST is the existence of resources (sources of specific information), each of which is referenced with a global identifier (e.g., a URI in HTTP). In order to manipulate these resources, clients and servers communicate via a standardized interface (e.g., HTTP) and exchange representations of these resources (the actual documents conveying the information). Any number of connectors (e.g., clients, servers, caches, tunnels, etc.) can mediate the request, but each does so without "seeing past" its own request (referred to as "layering", another constraint of REST and a common principle in many other parts of information and networking architecture). Thus, an application can interact with a resource by knowing two things: the identifier of the resource, and the action required—it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the server actually holding the information. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML, XML or JSON document, although it may be an image, plain text, or any other content.

REST uses HTTP to create the call request and receive the response. There are no standards to guide the way that call parameters are presented and, likewise, no guidance on how response parameters should be laid out (REST is an architectural approach, not a standard). Hence everything is freeform and it is up to the documentation to clearly articulate how the interface works. Anything but the simplest interface becomes prone to error. A browser can be used to quickly test a REST interface; this makes it possible to use a trial-and-error approach to working out how a REST interface should work. However, where authentication of users is required, this is often arranged around session state. Session state is not so easily manipulated within a browser, thus an authenticated REST interface can not easily be tested via the browser. Clearly, with this level of variability REST becomes complex for all but the simplest of use cases. This is demonstrated by the approach taken by both Google and Yahoo for their REST services. A developer toolkit is provided by both to ease the task of the developer.

The suggestion that REST is purely a reaction to the complexity of SOAP is an oversimplification. In some respects REST is a step backwards, since there is no self describing feature and it is the responsibility of the developer to create a call to the RESTful service. This involves creating a HTTP request with the call parameters added manually. The response can be anything, unstructured data, xml structured data etc.

One of the selling points of REST is that it does not need an XML parser to compose input parameters and process responses. This allows a RESTful service to be called from within a web page running JavaScript. A form of response mark-up known as JSON has been developed allowing a RESTful service to return a structured data response similar to XML which can be parsed with JavaScript.

The use of REST based Web services is gaining traction, for instance Google only offers a REST-based interface to all their machine based API, the switch from SOAP-based Web services taking place Dec 2006.

4.2.5.2 *Service Creation Standards Assessment Conclusions*

The history of distributed computing is littered with failed attempts to provide a simple set of technologies that could allow organizations to deploy software-based services that could easily be consumed by other software entities over a network. Within the last ten years, however two developments have given renewed hope to this vision:

1. The development, standardization, and roll-out of the Internet and Internet-based protocols (IP, TCP, HTTP, etc) which have now become ubiquitous.
2. The development, standardization, and roll-out of the basic Web Services specifications (XML, SOAP, WSDL) which have been implemented with a high degree of interoperability.

The use of these technologies allows simple Web Services to be exposed and consumed over the Internet and they represent the first true example of a widespread, open, distributed computing environment. They also represent the kinds of technologies that support the adoption of service-oriented networks. Internet protocols provide the networking fundamentals and Web Services provide the ability for CSPs to expose their services over those networks. However, these simply provide a basic set of technologies with which to deploy SON. For robust, reliable, scalable SONs a service provider needs further additions to these basics that deal with critical aspects of service to do with addressing, routing, trust, security, orchestration, etc. For these, there is no single solution and the routes to SON adoption broadly fall into one of two categories:

1. To continue to embrace the principles behind the use of Web Services and those of the basic technologies such as SOAP and WSDL, one must use a wide range of

extension technologies, which have poor levels of platform and language independence and therefore poor levels of interoperability.

2. To provide a controlled environment such as an SDK in which the additional requirements such as addressing and security are provided by the deployment itself.

Neither of these two options provides an open SON environment. SON deployments based upon WS-* may give the impression of openness but third party service providers will need to adopt the precise variants of the specifications employed in the SON. Moreover, the only way for SONs to interoperate will be to develop costly 'translation' code between the different environments – a backward step away from open services. On the other hand, SON deployments based around SDKs will only be able to provide a distributed services environment within the confines of the specific SDK deployment. Again, openness and interoperability will suffer.

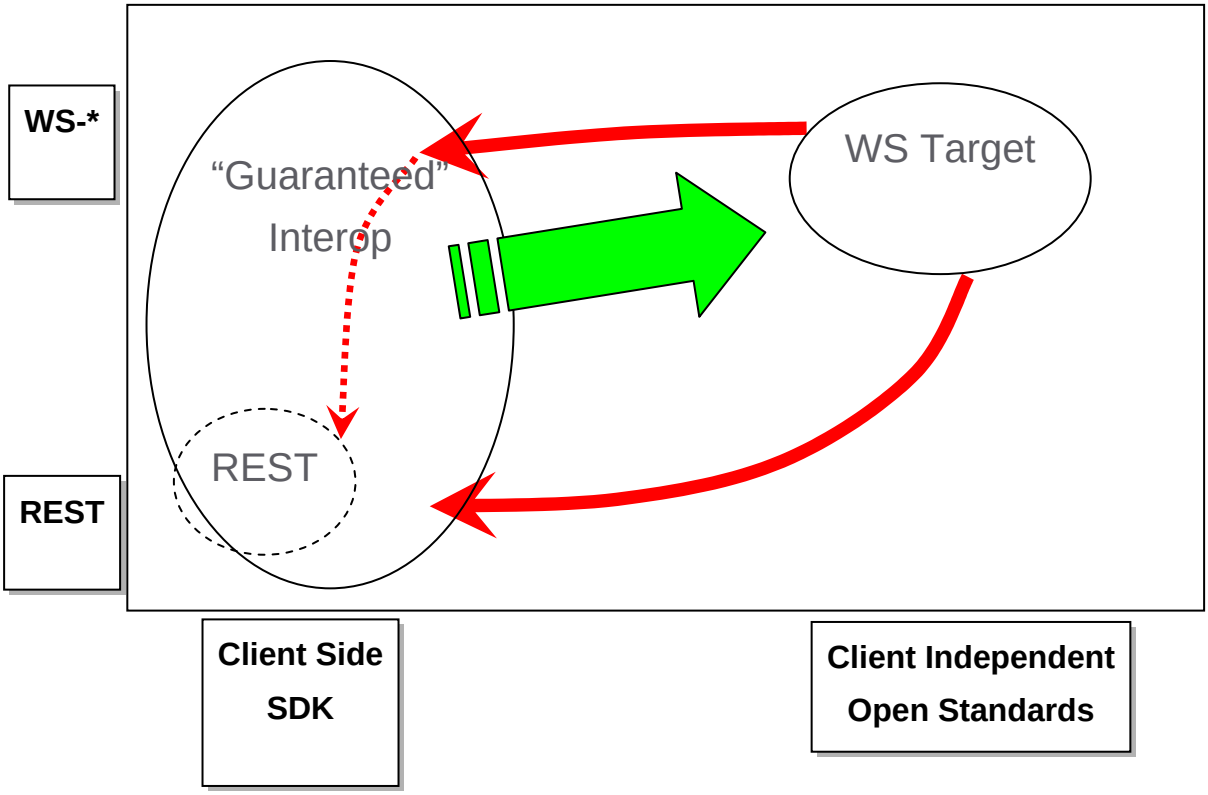


Figure 4-14: Web Services Standards Landscape

This figure shows an abstract representation of the Web Services landscape, and in particular how Web Services Extensions (WS-*) relates to REST. A number of key messages emerge from this picture:

1. This figure is based on the assumption that the target environment for ATIS member companies is based on an open, multi-vendor, extensible environment that will allow carriers to deploy a scalable, extensible SOA environment that encompasses a variety of vendors and third party service developers. It is believed that these attributes are

valuable to ATIS members. Our analysis suggests the best way to realize this is through a fully standardized web services extension environment.

2. In practice, the WS-* environment does not achieve these target values today, in part because WS extensions are currently inconsistent or even incompatible. This means that interoperability between carriers becomes very difficult. Some carriers have therefore decided they must adopt a specific subset of WS extensions, and provide a client side SDK to ensure new applications developed by third party developers will be interoperable with their SOA environment. This is represented by the thick arrow moving to the left from the WS target.

3. The deployment of a client specific SDK undermines the objective of a standardized client independent environment. The fact that one of the key potential values of WS-* (i.e. fully open, client independent services) has been compromised, has led some carriers to conclude that the complexity of WS is no longer bringing any value, leading them to move to a less complex SOA environment. This is represented by the thin dashed line moving down the left side of the diagram.

   In addition many third party developers have concentrated on services that are easily invoked with thin client applications such as a web browser (particularly on mobile devices). An interoperable web services stack is not readily available for many client service environments and therefore developers have sought an easier model based upon simple invocations based on HTTP. As a result of this other technologies have emerged such as those based on the REST architecture including technologies such as AJAX.

4. The arrow moving down and to the left from the WS target environment indicates the scenario where REST is adopted. The REST architecture does provide a degree of client independence for browser based applications (provided that the right version and plug-in support is available). However, it requires an SDK that fully defines the services characteristics for the developer.

5. The analysis so far shows that a variety of tactical pressures are pushing current deployments toward client specific service instantiations to enhance interoperability. However, from a strategic perspective, there is still (future) value in moving carrier deployments toward a fully open, client independent environment promised by WS-*. This is represented by the thick arrow moving back to the upper right quadrant of this diagram. This analysis suggests that an important role for ATIS is to facilitate this migration back to the target WS environment when the technology is sufficiently mature.

6. Based on the above, this analysis further suggests three key roles for ATIS moving forward. These are:

   - Work with selected SDOs to specify a profile for WS extensions and associated technologies (e.g. data models) that will ensure multi vendor interoperability in a WS-* environment. This could also include best current practices (BCP) guidelines to facilitate the early availability of the WS-* target environment.

   - Provide near term deployment guidelines that will simplify the eventual migration from today's pragmatic deployments, back to target WS-* environment.

- Provide a forum to engage developers from the web environment to get a firm understanding of their requirements so that we can develop standards which are of value to both communities.

4.2.6 Service enablers

As described in *Appendix A: Definitions and Terms of Reference for SON*, at its core the concept of a SE is:

- a module of software that provides an interface (one or more) to other 'consuming' software and through which it provides functionality that may be made use of by that consuming software
- its existence and how to make use of the SE is published
- the SE may have other published interfaces for lifecycle management and for charging
- the SE may make use of other resources, including other SEs and applications in order for it to perform its function

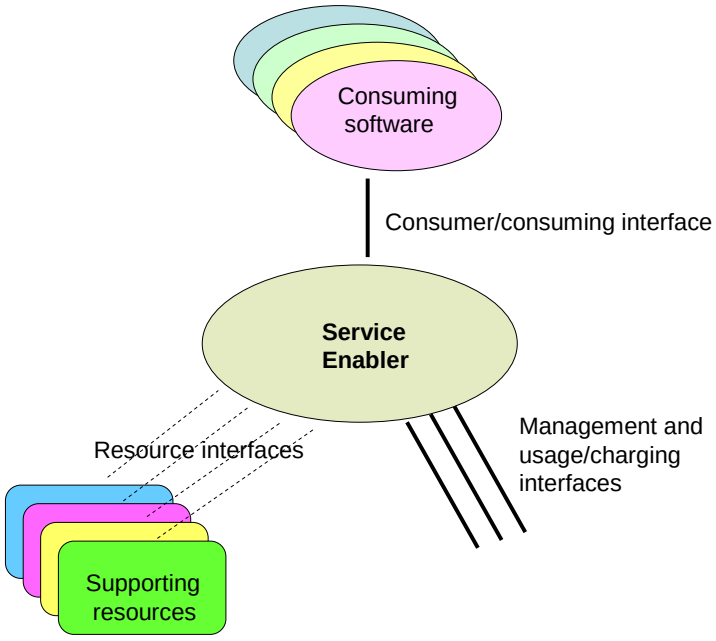


Figure 4-15: Service Enabler

The primary SE theme is software re-use. SEs may be created and deployed using a Service Delivery Platform. SEs may be for a network environment, an operations environment, a content provider environment or some other environment.

Note: As an extreme case, the SE functionality exposed to consuming software may in fact be all provided through the use of other underlying resources. In such a case, the SE software is reduced to a consumer interface, back-end interfaces to the other resources and supporting security, management, etc. capabilities. A Parlay X interface/gateway would be one such example. But more generally, an SE is viewed as including some or all of the functionality itself.

Historically, the SE general idea has been around and implemented in various guises for decades, e.g., Web Services, SOA, CORBA, Object-Oriented Programming technologies. OMA has defined over 30 'SEs', primarily driven by driven by wireless end-user services. Commercial companies offer SE products; e.g., IBM[4] and Hewlett-Packard [HP].[5] The Institute for Open Communications Systems (FOKUS) Open SOA Telco Playground has created a few 'SEs' that are available for use.[6]

Currently, there is no standardization of SEs, although existing "SEs" typically make extensive use of standards. Potential areas to consider standardizing are –

a. The set of basic SE consuming interface types.

b. The set of management and charging/usage interfaces. The TeleManagement Forum is working in this area.

c. The deployment/publishing mechanism. Published SE information minimally includes what the consuming interface is and what functionality is provided, and how/where to access the SE. Other types of information to consider including may be about, e.g., capacity, reliability and QoS. The format [metadata topic] of the information should be considered.

d. The functionality of particular SEs (e.g., a basic address book SE). Note that the OMA SEs or industry examples should be has defined over [30?] 'SEs', primarily driven by wireless end-user services considerations. Those should be examined for suitability.

e. Ways of restricting what the set of potential consuming software entities is allowed to be.

---

[4] http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&letternum=ENUS207-268

[5] http://h20247.www2.hp.com/enterprise/downloads/MessagingGateway_SolutionNote2.pdf

[6] http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_soa_telco_playground/research/service_enabler/index.html

f. Brokering/orchestration/coordination among SEs, akin to the same concept among service applications.

Standardization must support cross-domain scenarios – use of SEs by consuming software in another domain, and publishing of SEs so they are visible to other domains. An especial cross-domain concern is security.

4.3 3rd Party Interfaces

4.3.1 Introduction

The discussion about third party interfaces was bounded to the interfaces that telecommunication providers utilize to expose resources that they want to make available to interested parties. In this context, Parlay emerged as the most appropriate for Telco requirements and as covering the widest breath of services offered by an operator.

The discussions that follow highlights the importance of OSA Parlay and Parlay-X as mechanisms for abstracting the functionality of services supported by operators independent of the specific internal protocols needed to support such services.

A quick tour of other standards bodies highlights interesting work done by OMA in the area of Mobile Web Services. This work allowed development of requirements for the OMA Web Services Enabler Release.

The Gap Analysis and Recommendations section challenges us with three key questions:

1. What should Parlay's role be in the Service Oriented Network of the future?
2. Should telecommunication standards bodies such as OMA continue to suggest/recommend new and at times overlapping interfaces/architectures dealing with third-part interfaces?
3. What are the implications of Parlay-X and Web Services being supported from the device (handset or laptop)?

4.3.2 A Definition and Context

In the following discussion interfaces will be defined in the context of enablers. More specifically: *"An interface represents a means of exposing the function(s) of an enabler for use by any resource. A defined interface tells the resource what services the enabler that offer the interface is prepared to provide……Interfaces only tell a resource how the functions of the service enabler can be used. The interface makes no assumptions on the resources that may use it….."*[7]

[7] Source: "The Open Mobile Alliance – Delivering Service Enablers For Next-Generation Applications" by Michael Brenner, Musa Unmehopa

This discussion is further confined to the use of interfaces utilized in the telecommunication domain where the actors are network operators and third-parties. Network operators make available network resources and service capabilities to third-parties such as Application Service Providers in a secure, controlled and billable manner. The "northbound" *programmatic* interfaces that operators expose must conform to open standards, in order to support as many third-parties as possible.

One standards body that has made it its objective to define telecommunications specific interfaces is the "Parlay Group". *The **Parlay Group**[8] is a technical industry consortium (founded 1998) that specifies APIs for the telephone network. These APIs enable the creation of services by organizations both inside and outside of the traditional carrier environment.*

*"..**Parlay** is service architecture for controlled, manageable and billable third-party access to abstracted network service capabilities, and a suite of APIs to those network capabilities…Parlay X Web Services can be offered through a layered architecture on top of a Parlay Gateway that serves as a proxy toward the underlying network……"* as shown in Figure .

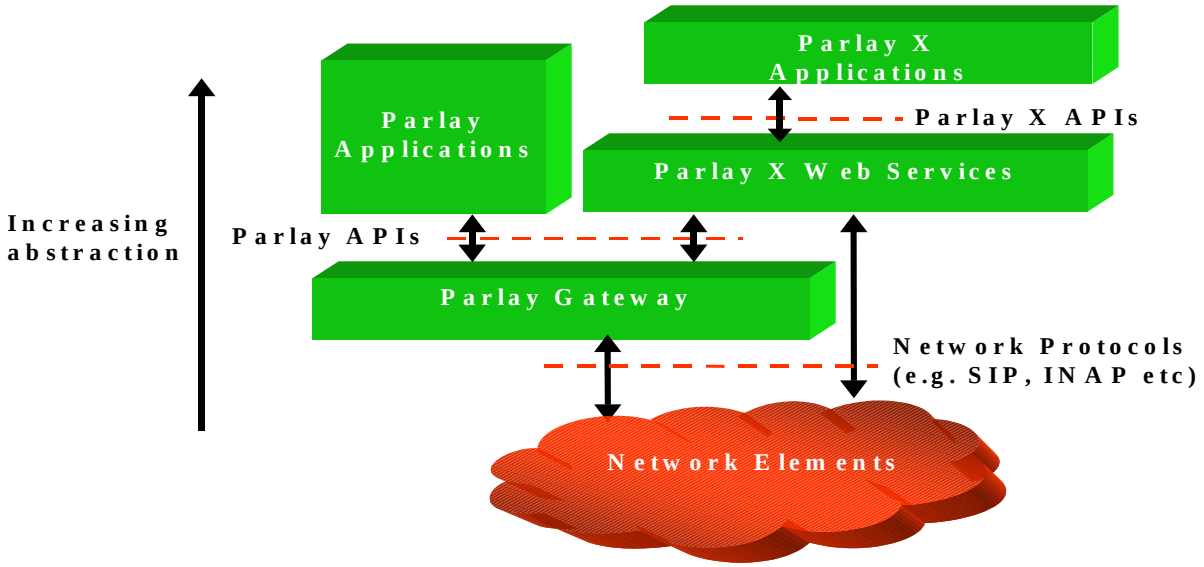


Figure 4-16: Parlay Architecture[9]

Parlay APIs (see Appendix C for a complete list) are implemented using any of the following realizations: OMG Interface Definition Language (IDL)/CORBA, W3C WSDL, and UML/Java 2 Platform Standard Edition (J2SE) and Remote Method Invocation (RMI) J2EE APIs. *"..Parlay X*

[8] Source Wikipedia

[9] Source "Parlay Group"

defines Web Services, that is, service capabilities deployable in a Web Services environment..". As such Parlay X supports service *creation*, *discovery* and *usage* via standard use of web services in the context of the Parlay *Framework* functionality:

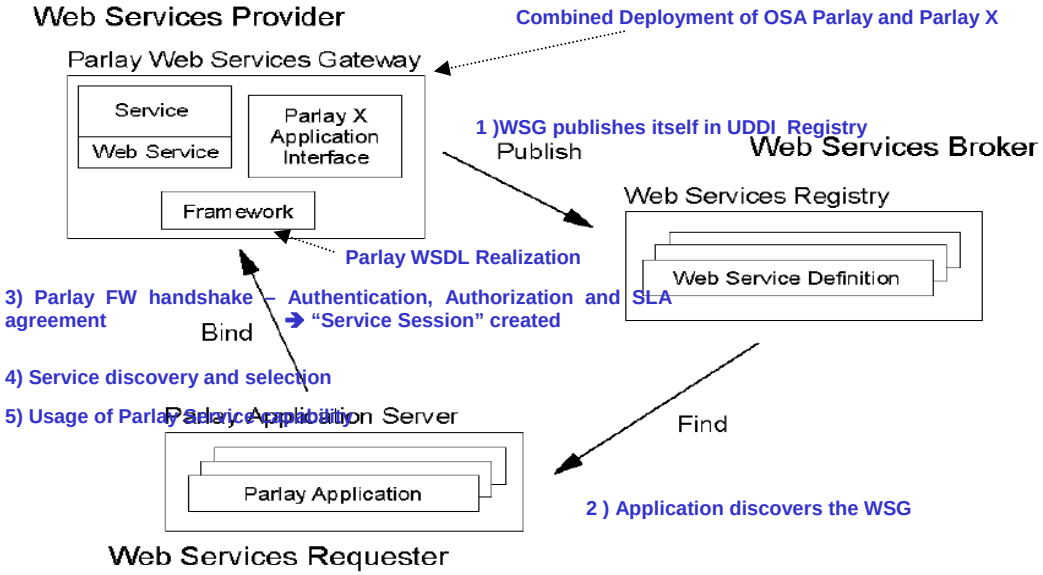


Figure 4-17: Third Party Web Services via Parlay X[10]

In September 2004 the specifications for Parlay X were submitted and approved as 3GPP TS 29.199 series specifications.

4.3.3 Parlay In Detail

In order to understand how Parlay supports a Service Oriented Network framework, it is critical to first gain an understanding of the inner-workings of Parlay. The goal of Parlay is to provide a means for end-user services (Client Applications) to gain access to the functionality (Service Capabilities) available in telecommunications networks.[11]

To allow the Client Application to get at the network functionality, Parlay defines a number of interfaces, each supporting a different capability of the network. **Each defined interface is known** as a **Service Capability Function** (or just "Service") which, when implemented is known as a **Service Capability Server** (SCS). [11]

[10] Source – "Parlay Web Services Application Deployment Infrastructure"

[11] Source "Parlay/OSA From Standards to Reality" Musa Unmahopa, Kumar Vemuri, Andy Bennet

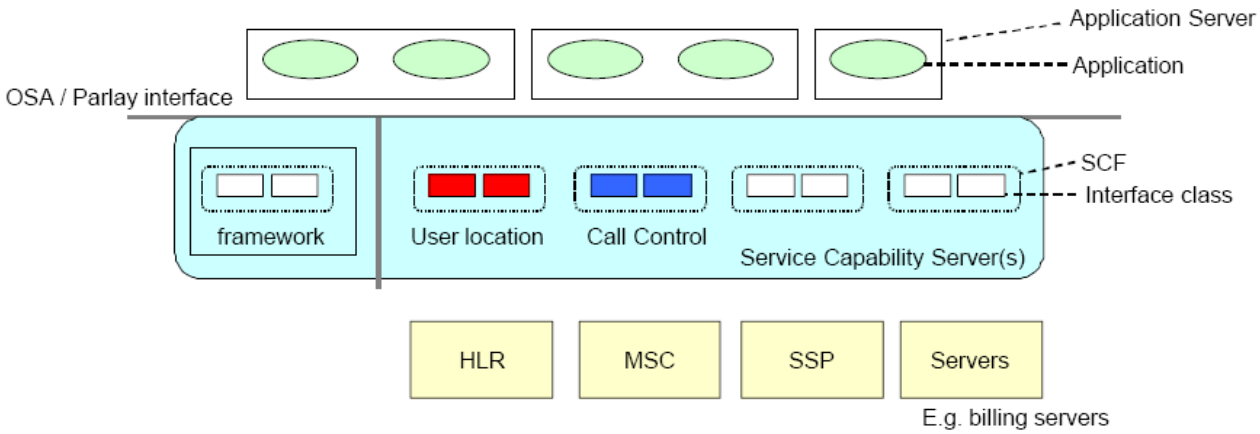


Figure 4-18: Parlay's Service Capability Server Architecture[9]

The Parlay Server Capability Server (SCS) provides:

- The role of "*mediator*" by translating requests from client applications into operations in the network
- The role of "*Policy Enforcement*" by defining what a client application is allowed to do in a given situation

Third Party applications that require services provided by the network can benefit from a *secure environment* provided by the Parlay "*Framework*" and Parlay SCS:

- The Parlay *Framework allows* client *applications* to *discover* operator services based on their *capabilities*.
- *Service Capability Servers* restrict client behavior by *defining* a set of *capabilities they support*
- *Registration* and *Discovery* procedures allow the Framework to put *Client Applications in touch with SCSs*
- The *Framework* and the *Client Application* performs *mutual authentication* using CHAP
- *Service Capability Servers* (SCS) can further use the *Policy Management* SCS to implement security functions such as *privacy* enforcement
- The *Client* Applications must electronically *sign* a "*service level agreement*" before being able to initiate a "Service Session" with an SCS

It is important to note that policy management in Parlay relates to policies at the service layer not at the subscriber layer. The Parlay architecture facilitates operator enforcement of policies on behalf of the application. The Policy Management Service Capability Server allows use of a *policy repository* in the operator network to *dynamically manage* centrally defined *network policy rules* "**if** *condition* ***then*** *action*" that can be evaluated by applications to provide consistent functionality across multiple applications:

- Clients can register for policy events
- Clients can be 3rd party applications as well as operator SCSs

- The Framework controls which clients can access what policies

Third Party applications also benefit from the high level of reliability expected from a *Telco grade* environment. The Framework with "***Integrity Management***" monitors the health of the "*service session*" between client application and Service Instance:

Load Management: How loaded (Central Processing Unit [CPU]/Memory) is the SCS?

Heartbeat Management used by Framework to *monitor health* of *service sessions* between Client applications and Service Instance

Fault Management:

1. Allows Client Application to notify Framework that it *cannot use a service instance*
2. Allows Client Applications to instigate "*Self Test*" requests to Service Instance
3. Allows Client Applications to *collect fault statistics* from a Service Instance

4.3.3.1 Third Party Interfaces using Parlay X - "Build them and they will come"

There is no formal identification of what a specific "unit of a service" represents in terms of worth of being exposed to third parties. From this point of view, Parlay is unique in its ability to provide third parties a view of what "*services*" operators could provide via a standard set of interfaces governed by a Telco grade *framework*.

Parlay defines the Service Capability Function (SCF) as an abstraction of a **unit of service** functionality available in a telecommunication network. It is a unit, in that SCF has a single main functional focus, for example, Presence, Location, and Charging. Each SCF supports an **Application Programming Interface** (API), the so-called Parlay API for the SCF.

There is a strong correlation between a service capability function and the concept of a "*service enabler*" in a Service Oriented Network. Recall that a service enabler was previously defined as a function (or set of closely related functions) in the domain of a service provider that is exposed though a defined interface, toward other resources. Service Enablers are re-used through their interfaces. There are two shared key elements between capability functions and service enablers:

1. Both are building blocks of re-usable functionality
2. Both expose their functionality via a set of well-defined interfaces

To attest to the extensibility of Parlay as a mechanism for exposing operator functionality to third parties it is important to note that the *Parlay framework* provides the same standardized mechanisms to publish, support discovery and participate in Integrity Management functionality for proprietary as well as standard based Parlay service capability functions. *Tables presenting Parlay X Service Descriptions and OSA Parlay APIs Descriptions can be found in Appendix C.*

4.3.4 *Parlay as a Service Enabler in a Service Oriented Network*

A possible way to implement service enablers in a service oriented network is by leveraging Parlay interfaces such as OSA Parlay and Parlay-X in support of the functionality provided by Parlay Service Capability Functions as shown 4-19.

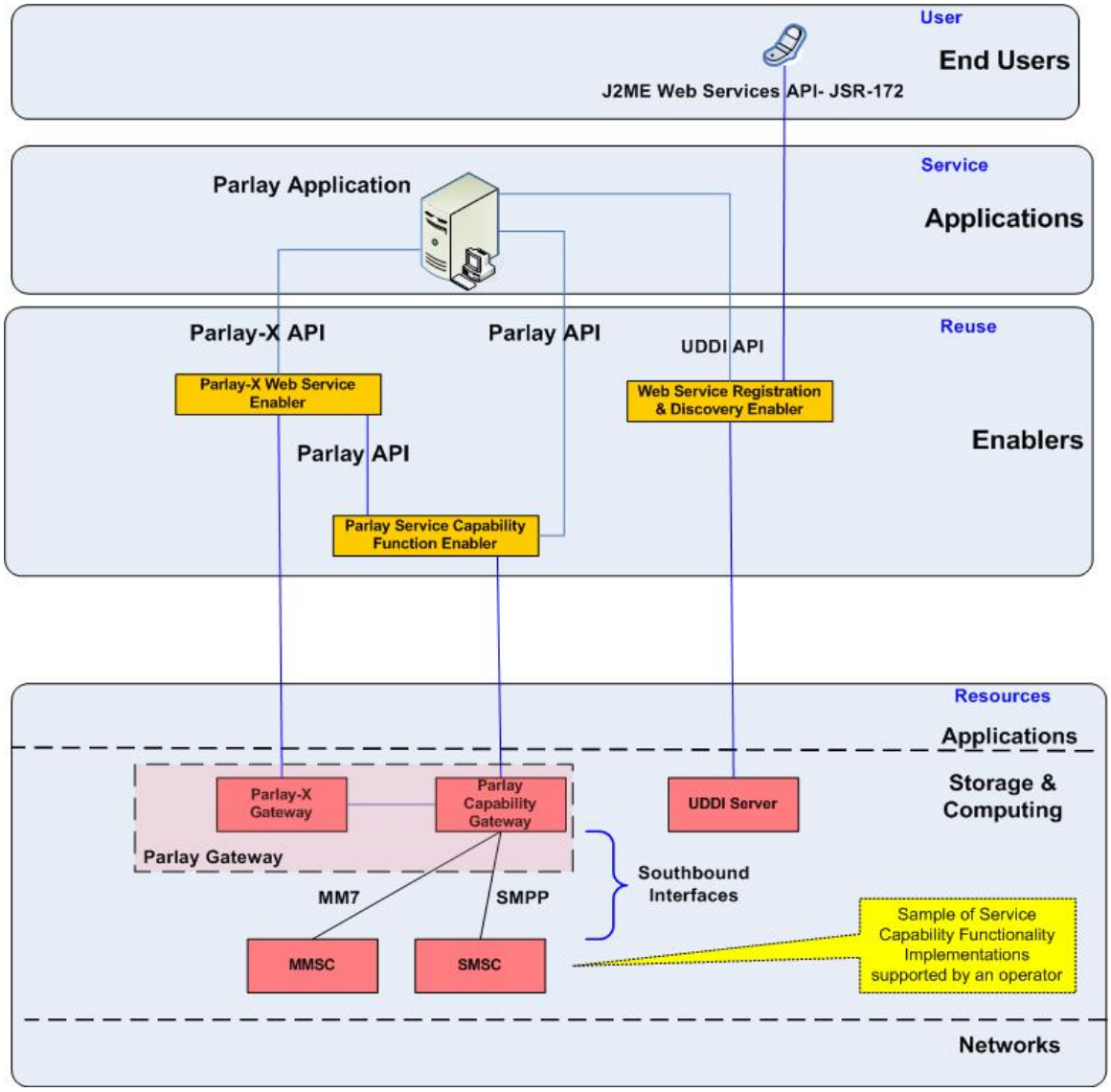


Figure 4-19: Parlay as a Service Enabler in a Service Oriented Network

Operators have a wide array of options for supporting service enablers via Parlay when one considers the breadth of service capabilities listed in Appendix C).

4.3.5 Why Emphasize Web Services Interfaces over other types interfaces?

The "OMA Web Services Enabler (OWSER): Overview" (OMA Web Services Enabler (OWSER): Overview)" provides a compelling explanation for why a Service Oriented Architecture and its supported Web Services interfaces are superior to previous architectures and interfaces (such as OSF DCE, Microsoft's Distributed Component Object Model (DCOM), OMG's CORBA, and Java RMI):

1. **Messages versus Application Programming Interfaces (API):** In a SOA, the interaction between a service and its consumer is based on conversational, message-based interactions. That is, the requester and the provider of a service exchange messages, conforming to a particular protocol and data description syntax (more formally, a schema). In contrast to this, traditional distributed computing techniques like CORBA, DCOM, and Java RMI have been based on (language dependent or language-independent) APIs, where the paramount goal is to provide a programmatic model of the interaction between a service requester and a service provider.
2. **Coarse-grained versus fine-grained interfaces:** APIs, by definition, deal with programmatic interfaces and hence programming level constructs, i.e., programming objects, and their methods. A SOA, by contrast, does not deal with programming constructs, but with services that offer a different granularity for their interface. Such services are defined not by any underlying programming objects but by the facet of the business that the service exposes, e.g., submitting an invoice to a purchasing system, updating a travel reservation in a calendar application etc. Such interfaces offer descriptions of what messages are suitable input and output to and from the service, and under what conditions they are exchanged. How messages are processed or generated are deliberately out-of-scope of a SOA, and viewed as an implementation matter. Another way of stating this difference is to say that the coarse-grained interfaces of SOAs are more like contracts (a description of what an offering party is willing to do under what conditions) rather than the more traditional programming interface description as in CORBA or DCOM.
3. **Asynchronous versus synchronous interactions:** Such conversational exchanges in a SOA are defined to be asynchronous, in that there is no predictable timing relationship between the exchange of messages, that responses may be delivered long after a request was sent, and responses may be received on different transport connections than the one on which a request was sent. This contrasts with most programmatic interactions, which are synchronous by definition.
4. **Peer-to-peer versus client-server interactions:** SOAs are geared towards application integration where two independent applications exchange well-defined data and messages to complete some task at hand. API driven distributed computing are invariably client-server interactions, which in itself is a programming construct,

where there exists a pre-defined and a very well defined coupling between the two sides of the distributed application.

5. **Interoperability versus application portability:** Entities in a SOA interact through well-defined (or extensible) messages and the aim is to ensure that the two peer applications can support the message semantics and syntax in completing their independent tasks. By contrast, much of the work in traditional distributed computing has been to ensure both client-side and server-side application portability across heterogeneous languages and platforms through standardized language bindings (for the client side) and standardized object adapters (for the server side).
6. **Discovered partners versus pre-defined partners:** In its most general usage, a SOA allows parties to discover useful and compatible services. In practice, however, at least in terms of early adoption of this concept, most interacting parties are typically already known to each other, and some sort of prior business relationship governs their partnership. However, typical programmatic interactions in traditional distributed computing are not based on such a form of service discovery, but rather between well-defined entities.
7. **Internet versus intranet usage:** SOAs, which not restricted to interactions between business partners that take place across the public Internet, offer much of their advantages in such an environment. In contrast, the traditional distributed computing technologies have primarily been restricted to applications in an intranet environment owing to difficulties in scalability and firewall traversal when used across the Internet.

4.3.6 Other Standards

Standards such as OASIS or World-Wide Web Consortium (W3C) do not define web services interfaces for the telecommunications community. OASIS however, has just started "*The OASIS Telecommunications Services Member Section* (OASIS Telecom)" activity which promises to bring the full advantages of SOA to the telecommunications industry. OMA on the other hand, does target the telecommunication sector. OMA reconciles Parlay to the OMA Open Service Environment as shown in Figure 4-20.

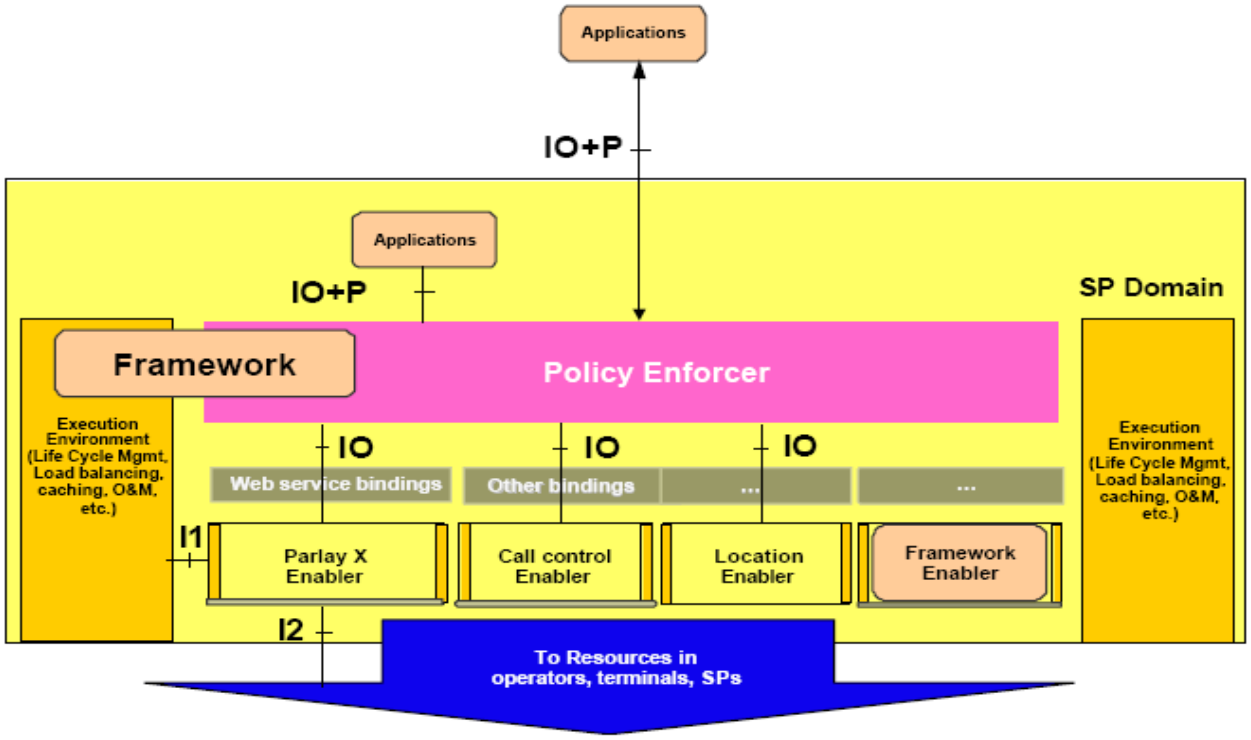


Figure 4-20: Parlay in OMA OSE Architecture

The OMA Mobile Web Services Working Group has also closed the activities associated with the OMA Web Services Enabler Release (OWSER). This work has moved web services into "*Mobile Web Services*". A couple of novel ideas[12] have emerged from this work:

1. ***The Mobile Device as a Web Service Requestor***. Figure 4-21 shows how a handset would access web services in an operator network using JSR 172.
2. ***The Mobile Device as a Web Service Provider***. The web service is actually offered by the mobile and discovered and consumed by any web service requestor. An example of a service accessible from the device could be the address book.

4.3.6.1 Mobile Web Services-OSA and J2ME

The idea that a mobile can act as a *web service requestor* is not new and has been acknowledged in the OMA Web Services Enabler. OMA recognizes two architectures for supporting the mobile device as a web service requestors as shown in Figure and has adopted the "*Direct Model*" where a mobile device, supporting a web service stack, would act as a web service requestor directly to the Web Service Server.

[12] See "Mobile Web Services – Architectures and Implementation" by Frederick Hirsh, John Kemp, Jani IIkka

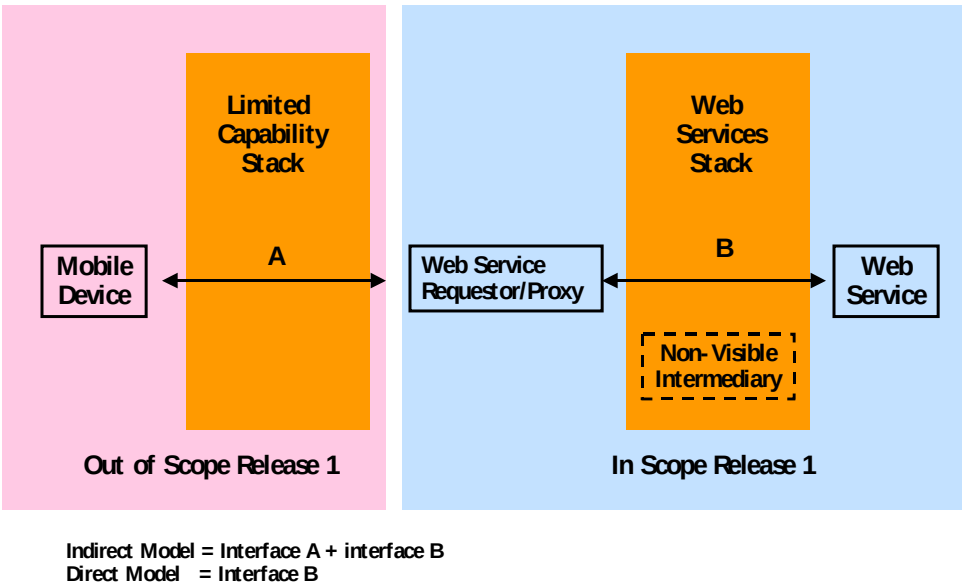


Figure 4-21: Mobile Web Services Architecture Models

Consider applying the "Direct Model" to a "Parlay-X" client application supported on a mobile device or a desktop. A paper titled "*OSA/Parlay and Java 2 Platform Micro Edition (J2ME) in Tandem: Developing Innovative Services by Combining Intelligence of the Application Server and the Mobile Terminal*" by *J. Domaszewicz, M. Rój, M. Kunikowski* describes an implementation of a "Parlay-X J2ME Mobile Information Device toolkit (MIDlet)".

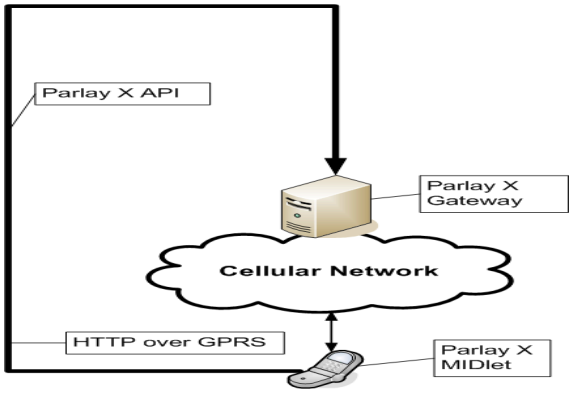


Figure 4-22: Parlay-X MIDlet Based Architecture

The authors explain why the "direct model" yields superior results versus and "indirect model". In their words:

> "*One way to enable MIDlet to access telecom network functionality (as exposed by the OSA/Parlay APIs) is to implement an OSA/Parlay Proxy application, as described above.* ***A*** *drawback of that approach is the need to use a proprietary MIDlet-to-Proxy protocol.* ***Another, more direct and open way to achieve the same goal is to use Parlay X and make the MIDlet itself a Parlay X application (we call it a Parlay X MIDlet).*** *With the current level of resources (especially memory) available in high-end mobile phones, a MIDlet can handle the Web services protocols. Due to" potential problems with preserving the IP address of the*

terminal (maintaining a General Packet Radio Service [GPRS] session), the preferred Parlay X APIs are those, for which the MIDlet acts as a Web services client, and the gateway acts as a Web services server. Fortunately, most Parlay X APIs fall into this category."

The key point is that Parlay-X, deployed in the way described, could provide a ***standard based architecture*** that is deployable both on Application Servers and devices (mobile devices and desktops/laptops) in an operator network. An OSA Parlay Application Server or a Parlay-X MIDlet would use the same Web Services protocol to access operator services. ***Consider how much easier it would be for an operator to support policies across devices and application servers, since the same interfaces and protocols are used.*** Figure shows how the "Direct Model" architecture for supporting Parlay-X web services requests can be supported in the Service Oriented Network ATIS framework.

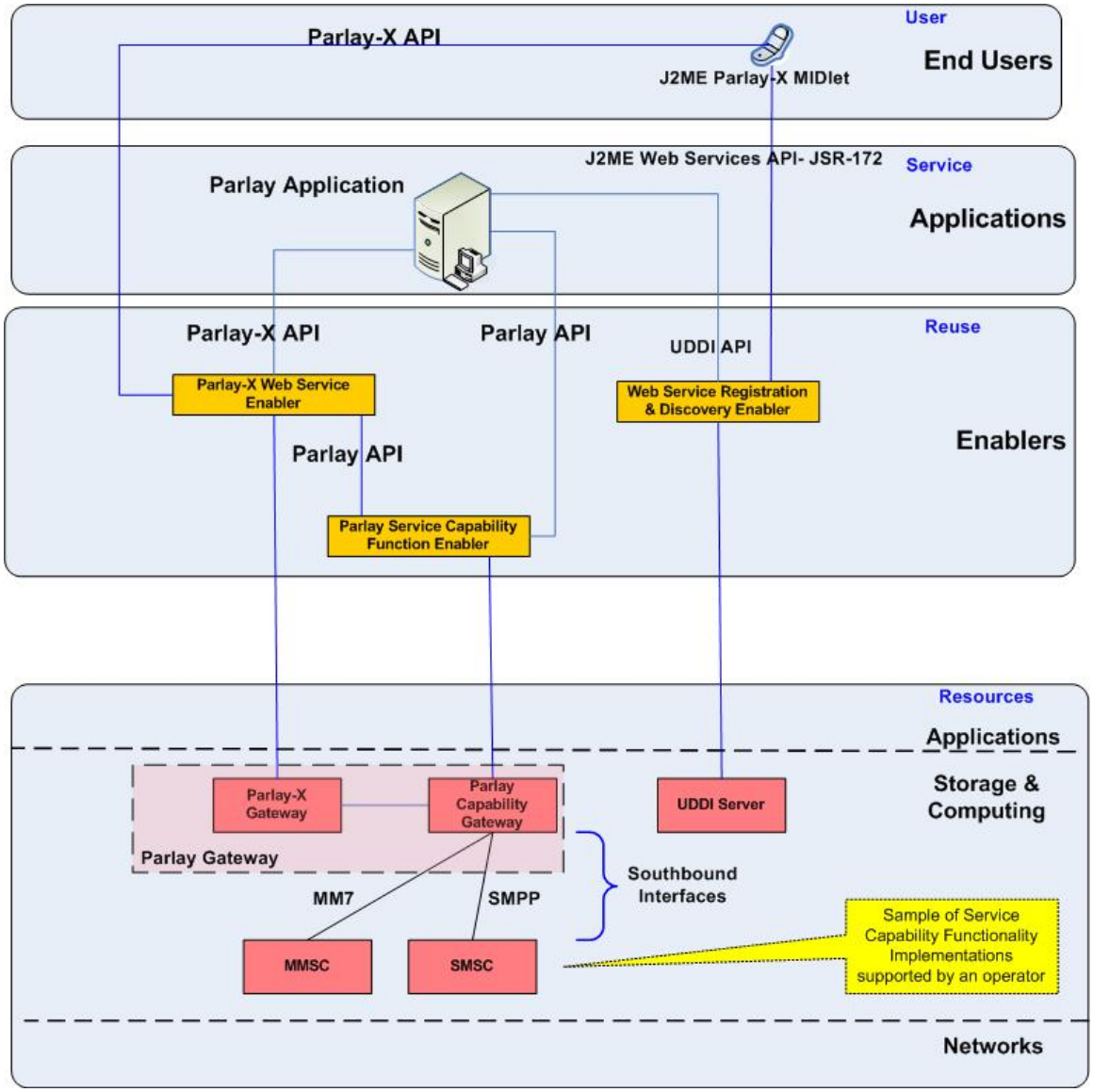


Figure 4-23: Parlay-X as a Mobile Web Service Enabler in a SON

4.3.7 Identified Gaps

After exploring ways to provide access for third parties to telecom resources via specific interfaces, it is apparent that Parlay-X today represents the most extensive and powerful architecture to support Telco grade interfaces. The discussion on Parlay-X has shown how web services interfaces can be extended both in standards and non-standard ways by leveraging the powerful capabilities of the "*Parlay Framework*".

The Service Oriented Networks of the future will quite likely leverage IMS architectures. As of now, applications on the device that access services in the operator network will be able to leverage the full power of the IMS control plane ***if*** they can use the "SIP". IMS refers to the implementation of these services (e.g. Session Initiation Protocol for Instant Messaging and Leveraging Extentions (SIMPLE) Presence) as SIP Application Servers. As one scans the Parlay interfaces in Appendix C, it becomes apparent that Parlay is abstracting the "services enablers" supported within the operator network independently of the signaling plane used to operate these enablers. We also have seen that OMA has acknowledged "Mobile devices as Web Service Requestors". ***What is missing is an acknowledgement that a device supporting Parlay-X directly on the device or perhaps via a Parlay-X Proxy Gateway would allow applications to access operator resource in a way that is not possible by utilizing the session initiation protocol alone***.

As we acknowledge the possibilities available by accessing operator services via the established Parlay-X interface from both application servers and handsets, then one wonders why other interfaces and architecture are being sought after that provide similar functionality. For instance, OMA fulfilled the Mobile Web Services Network Identity requirements by acknowledging the Liberty Alliance Identity Federation Framework (ID-FF). Two concepts were introduced:

1. Identity Federation
2. Attribute Sharing

Attribute sharing is a process where the identity attributes (e.g. location or presence information) of a subscriber (principal) can be queried or modified. Parlay-X supports interfaces capable of retrieving identity attributes such as location or presence via the "*Terminal Location*" and "*Terminal Status*" interfaces. We must consider:

- *What additional value does the Liberty Alliance ID-FF bring to bear that Parlay-X does not?*
- *Should the ID-FF architecture be deployed alongside Parlay architectures?*
- *Is there a need for to interoperate between Liberty Alliance ID-FF and Parlay architectures in areas of overlapping functionality as is the case for "attribute sharing"?*

The functionality covered by "Identity Federation" is not covered in Parlay. Identity Federation is a process of associating identity data about a Principal that exists in two or

more entities. An immediate result is the ability to provide a service called single sign-on, which allows a Principal to authenticate once (at a trusted Identity Provider), rather than every time it accesses services at other Service Providers, within an active authentication session.

- ***Should this functionality be provided by extending Parlay-X? Or is Parlay-X not the appropriate set of interfaces to support Identity Federation type functionality?***
- ***What should Parlay's role be in the Service Oriented Network of the future?***

4.3.8 Recommendations

Recommendation removed for external transmittal.

4.4 Service Quality Management

Service Quality Management (SQM) is a sub-set of service management that is concerned with the quality of the service being provided. It is related to the area of service-level agreements and service-level guarantees in that these provide the measures and contracts against which SQM is assessed. Ultimately, SQM is concerned with the level of quality that the service user perceives – this is often termed QoE.

With traditional telecommunication services such as residential voice, SQM has typically been entirely contained within the domain of the service provider. The end-to-end network is controlled to ensure a high voice call quality and many telecommunication licensing organizations set standards for quality that the provider must adhere to. In many cases, the provider only guarantees the service to the network termination point at the first point of entry into the premise. In this way it can ensure that it has control over all aspects of service delivery and can therefore maintain service quality levels.

In a world of service-oriented networks, service providers will be integrating products using components from a wide variety of sources, many outside their own domains. In order to be able to guarantee a level of QoE to the user, the provider must address SQM at all points in the value network. In today's multi-media, multi-technology, multi-provider world, the level of control a provider has over the QoE is less deterministic. Services such as IPTV may involve a complex value network, involving content providers, content aggregators, advertising providers, network providers, home gateways, home networks, a wide variety of user devices – all of which providers the IPTV service provider with a multi-layered set of service quality domains that must be able to be related to the overall QoE, see Figure 4-24.

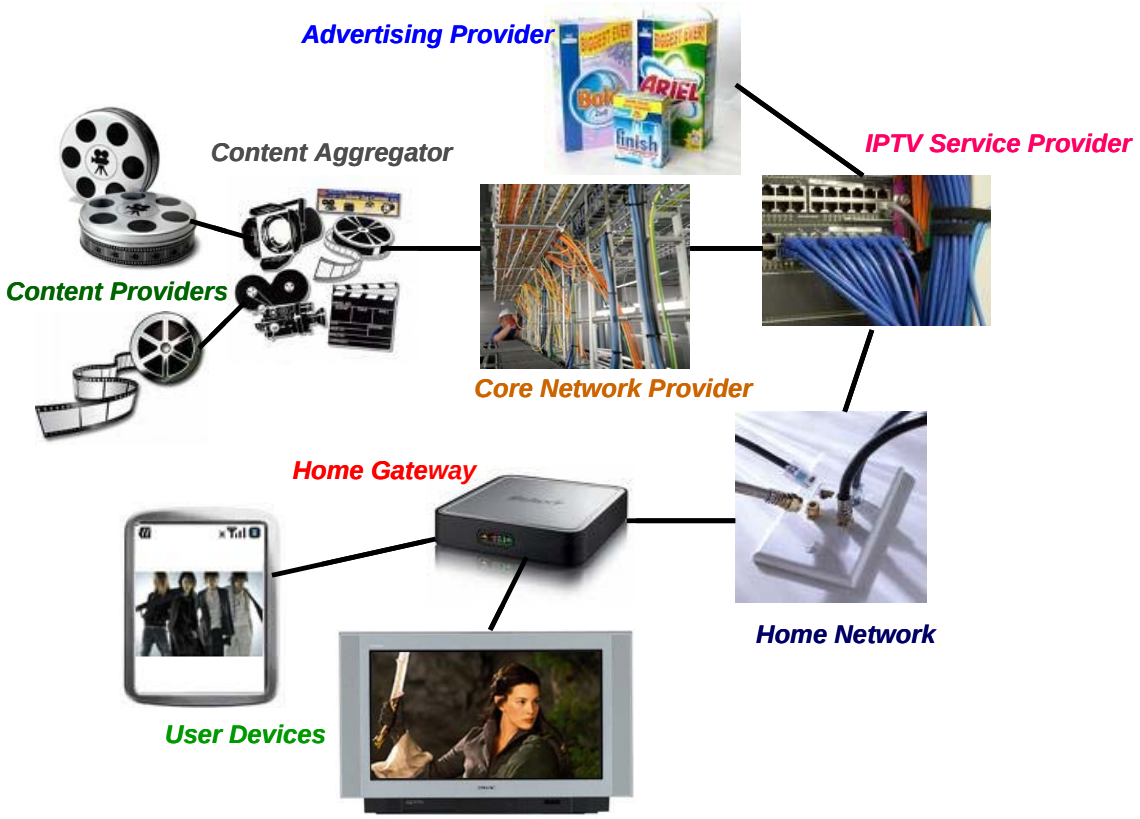


Figure 4-24: IPTV Value Network

Consider the scenario of the user watching an IPTV movie and the picture suddenly becomes fuzzy and un-watchable. The following questions arise:

- Where is the problem located – is it the content, the applications, the network, the equipment?
- Are there any interactions that suggest the problem isn't in one domain?
- Whose responsibility is it to resolve the problem?
- Are the any service-level agreements broken?
- What is the prognosis for the user – how do they determine this?

These are the types of questions that service quality management must help answer. For service providers, SQM is an increasingly important subject area. In particular, service providers offering IPTV services are competing with broadcast TV and with the use of DVDs and other kinds of portable media. These offer a high level of viewing quality which has set the user expectation high.

Note: the IPTV example above highlights the SQM issues related to the user's service experience. Related to this however, will be the user's experience in the overall service wrap. That is, the aspects of the service concerned with ordering, fault management, billing, etc. When a user orders an IPTV service there will be a complex value network that will be affected in terms of fulfilling the order. The fulfillment of each part of the order by providers in the value network may also have service level agreements, for example a provider may agree to establish

a user's account and turn on the streaming of movies within 24 hours of the order being accepted. Therefore, QoE and SQM are to be applied to the service and to the service wrap. The industry will seek common solutions to these areas.

4.4.1 SQM and Business Models

The IPTV scenario illustrated in figure 4-24 shows an extended value network with many participating providers. As with many services, at each leg in the value network, the business model involves the provision of service from one business entity to another. There will be a contract between customer and supplier, with service provided in one direction and a revenue stream flowing in the opposite direction. Each revenue stream is associated with an SLA which may be explicit, implicit or customer-defined. In the case of a customer-defined SLA this may involve complex heuristics and algorithms which are an area of competitive differentiation for service providers.

As service providers adopt practices from the world-wide web, we are seeing advertising play an increasing role in the overall service offering. In the content industry we see some participants in the value network adopting two-sided business models in terms of a) the provision of advertising services to businesses wishing to promote their goods and services, and b) subscription and usage-based services to individual users. These more complex forms of business exchange pose even greater challenges for SQM. For example, users will expect the subscribed service to meet the SLA, whereas advertises are more interested in those users making follow-on sales for other goods and services. The SLA between service provider and advertiser will have entirely different characteristics from the user SLA.

4.4.2 SQM Standards

The TeleManagement Forum (TMF) has recently established an SQM program to address this subject area. The program has outlines the following key needs to be assessed:

- Measuring Customer Satisfaction;
- Pinpointing problems across the Value Network / Ecosystem;
- Apportioning payments and maintaining security;
- Policing Service Level Agreements.

The TMF Board has mandated that the program defines at a minimum:

- A holistic framework for measuring and effectively managing service quality;
- Key service quality metrics at each point along the service delivery network;
- Service quality issues and the necessary accounting and rebating information; usage information, and problem resolution information;
- Management capabilities to support each step in the service delivery network;
- Appropriate interfaces /API's to enable the interchange of such information electronically between the various providers in a service value network.

The TMF's SQM program has recently reviewed and agreed its charter, including an outline workplan for the programs activities. The first phase will focus on getting the basic framework

metrics and issues identified and a solution path for improving Customer Experience established. There will be a Catalyst at the MW Orlando 2008 to show and demonstrate the key technical enablers for End-to-End (E2E) Service Quality Management.

4.4.3 Recommendation/Identified Gaps

Recommendation removed for external transmittal.

5 DATA MODELS, IT INFRASTRUCTURE ,OSS/BSS AND TRUST MODELS

5.1 Data Models

The SON will be a data rich environment. Today, work is occurring in domain specific environments. There are three high level categories of data model work: identity management, service specific schema definition, and modeling semantics. In reality, identity management is a service, but it is separated here because of the large amount of work in this area.

Standardization of data models is valuable where company to company interoperability is desired. There needs to be a global treatment across all services of the concept of who are you, where are you, what network are you accessing the services via. One example is identity management where customers roam from carrier to carrier. A second example is IPTV where multiple companies in the consumer, network, service, and content domains work together for the delivery of a single finished service.

Standardization of data models also reduces integration cost where data is widely used across multiple services. An example is contact information. Data model standardization of contacts eases integration across network databases, network services, and Customer Premises Equipment (CPE). All of these devices use contact information, but today they have proprietary means of storage.

5.1.1 3GPP IMS

IMS defines standard interfaces to be used by anyone to develop services. This standard schema defines how users are defined, relate to services, and how information is shared between services.

5.1.1.1 *User Identification*

- Subscriber: A Subscriber is an entity (comprising one or more users) that is engaged in a Subscription with a service provider. The subscriber is allowed to subscribe and unsubscribe services, to register a user or a list of user authorized to enjoy these services, and also to set the limits relative to the use that users make of these services.

- Private user id - A unique global identity defined by the Home Network Operator, which may be used within the home network to identify the user's subscription (e.g. IM service capability) from a network perspective. The Private User Identity identifies the subscription, not the user.
- Public user id - The Public User Identity/identities are used by any user for requesting communications to other users. For example, this might be included on a business card.
- IMS Service Profile - Collection of service and user related data defined and maintained in the HSS.

5.1.1.2 *Service Definition*

- Public Service Identities (PSIs) - An identity allocated to a service hosted in an Application Server
- IMS Prerequisites: IMS service provider has to authorize the end user to use the IMS service.

5.1.1.3 *Subscriber Data Model*

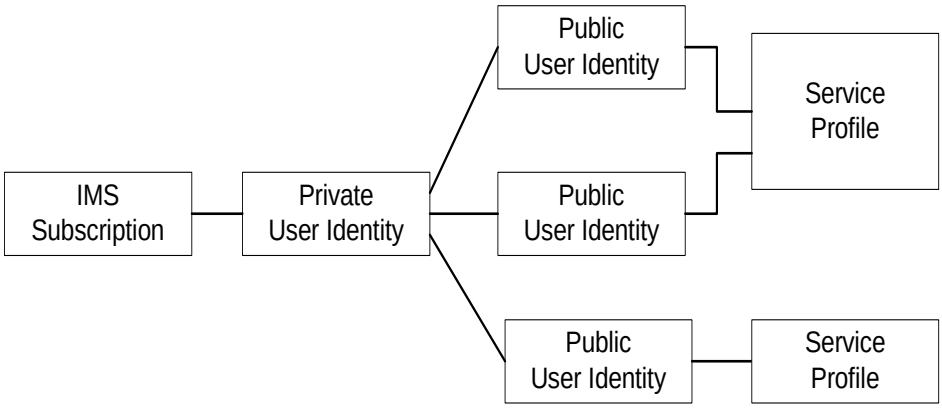


Figure 5-1: Subscriber Data Model

5.1.2 TM Forum Information Framework (SID)

The Shared Information/Data (SID) model is an abstract common model that is a central component of the TM Forum's Next Generation Operations Systems and Software (NGOSS) initiative. The goal of NGOSS is to promote open, distributed OSS/BSS systems using commercial off-the-shelf technology. Loosely-coupled systems require data abstraction, which the SID provides as an industry standard model. Reuse requires that the standard model also be abstract to enable different parts of a business to share data.

5.1.2.1 *User Identification*

- Party – A legal entity that can be sued. Contains attributes that describe individuals or organizations (i.e., name, address, phone, email, PartyId, etc.)
- PartyId - for service authentication
- PartyRoles – A Party can play many roles (employee, Customer, student, teacher).
- Customer – For extensibility, Customer is a subclass of PartyRole
- CustomerAccount – a subclass of Customer with billing information

5.1.2.2 *Service Definition*

- ProductSpecification – details of the product
- ProductOfferings – the way the service is offered to the market – this is what the customer buys
- Product – the way the service is maintained and performs while in use

5.1.2.3 *Party Data Model*

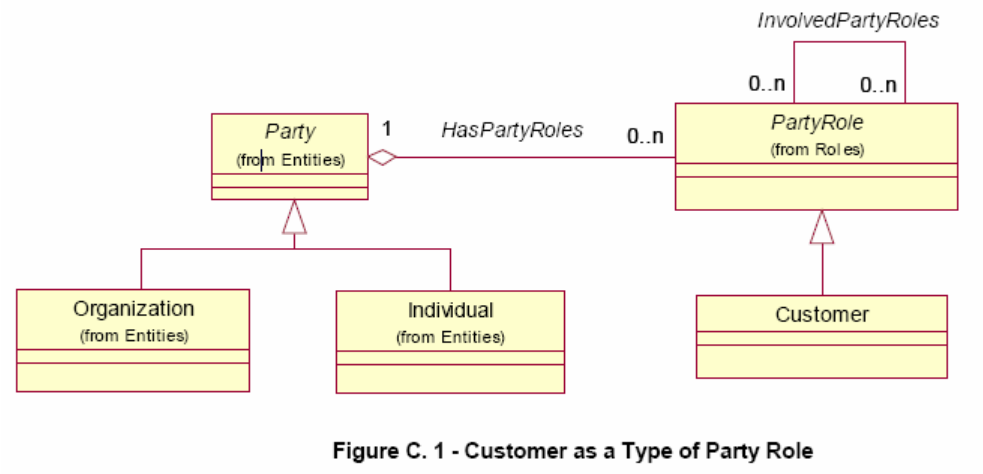


Figure C. 1 - Customer as a Type of Party Role

Figure 5-2: Party Data Model

5.1.3 Liberty Alliance

The Liberty Alliance was established with the goal of establishing an open standard for federated identity management. The Alliance has produced work on:

- Open standard and business guidelines for federated identity management spanning all network devices.
- Open and secure standard for Simplified Sign-On (SSO) with decentralized authentication and open authorization

5.1.3.1 *User Identification*

- A Principal is an entity that can acquire a federated identity, that is capable of making decisions, and to which authenticated actions are done on its behalf.
- An Identity Provider (IdP) is a Liberty-enabled entity that creates, maintains, and manages identity information for Principals and provides Principal authentication to other service providers within a circle of trust.
- A federated identity architecture delivers the benefit of simplified sign-on to users by granting rapid access to resources to which they have permission, but it does not require the user's personal information to be stored centrally.

5.1.3.2 *Service Definition*

Service definitions are completed by service providers and primarily designed for discrete services. Identity management service is shared.

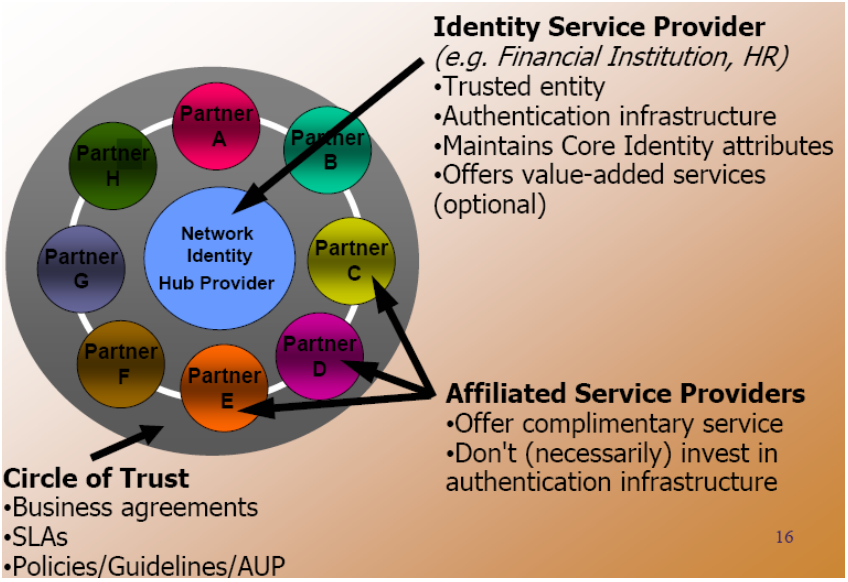


Figure 5-3: Liberty Alliance Service definition Model

5.1.3.3 *Subscriber Data Model*

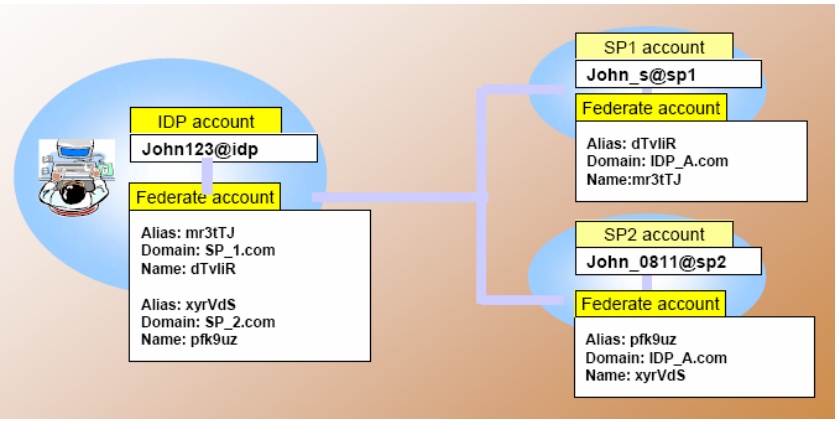


Figure 5-4: Liberty Alliance Subscriber Data Model

5.1.4 ITU-T IdM

The ITU-T model provides a framework for PCs, mobile phones, intelligent ID cards (e.g., electronics driver's licenses). The objective is to describe a framework to enable interoperability of existing and new applications that operate over diverse Identity Management (IdM) systems/networks.

5.1.4.1 *User Identification*

An Entity (user) has one or more Digital Identities with Relationships and Attributes. Policy attached to pieces of info (i.e. SS#) describing why the data is needed by the service. The user can control if a piece of info is shared.

5.1.4.2 *Subscriber Data Model*

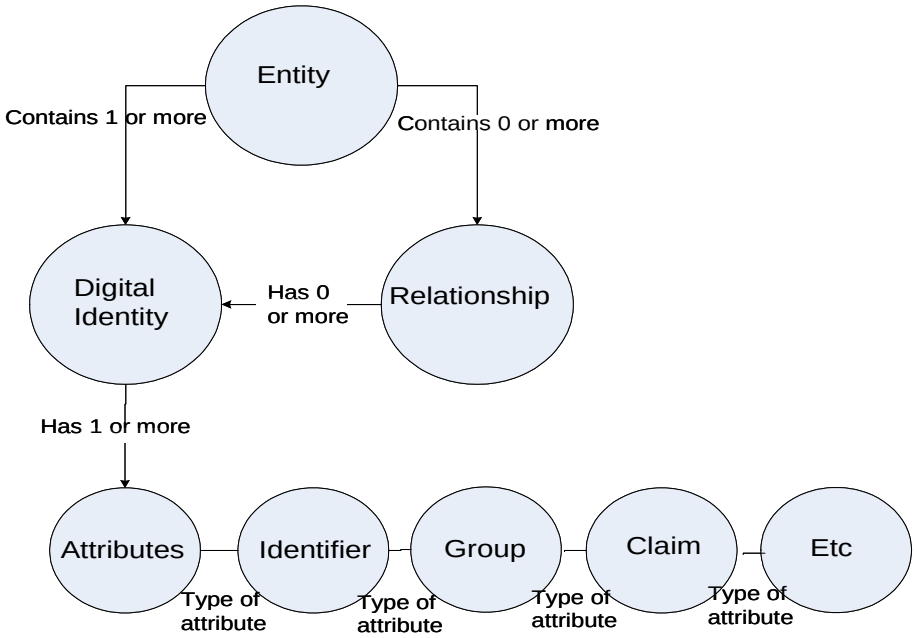


Figure 5-5 ATIS IPTV Interoperability Forum (IIF)[13] Service Specific Data Model Example

The ATIS IIF work is an example of service specific data model.

The business need is to document requirements for Consumer (Subscriber and User) profile and preferences metadata to support the following:

- User identification, Authentication and Authorization
- Identification of User capabilities
- User Customization or configuration of services
- Presence
- Notification preferences[14]
- Location preferences
- Policy management (for example, subscription period, limiting number of downloads, etc.)
- Parental control services
- Billing and settlement of transactions

[13] IPTV CONSUMER METADATA REQUIREMENTS, IIF-WT-027

[14] See ATIS-0800017 *Network Attachment and Initialization of Devices and Client Discovery of IPTV Services*

- Audience measurement
- Customer Care
- Advertising
- QoS / SLA management

In general, Consumer metadata specifies the user specific data required to configure, customize, operate, support, and bill an IPTV service.

The interoperability enabled by standardization of Consumer metadata will greatly reduce the effort required by all the relevant parties to provide IPTV services.

5.1.5 W3C Semantic Web Activity[15]

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF).

5.1.6 DMTF Common Information Model (CIM)

The Distributed Management Task Force, Inc. (DMTF) has completed significant work on virtualized computing environments. The Common Information Model is part of this work and provides relational structure to the virtual environment. In a traditional server, the disk drives, memory, processors, and other peripherals are physical entities that can be touched. Loading the operating system onto a traditional server is straightforward.

The virtualized environment is more complicated because it is a concatenation of logical resources: a section of disk, an assignment of memory, a time slice of processor, and mapped I/O interfaces. The DTMF CIM defines these relationships as well as the middle layers required to abstract the virtual system from the physical system. *CIM System Virtualization Model White Paper*[16] provides an excellent overview of these concepts. It also introduces the key relationships in the data model.

5.2 IT Infrastructure virtualization

[15] http://www.w3.org/2001/sw/

[16] http://www.dmtf.org/standards/published_documents/DSP2013_1.0.0.pdf

5.2.1 Overview

5.2.1.1 **Infrastructure**

IT Infrastructure can have different meanings dependent on the user and the context, but in general IT Infrastructure refers to the equipment, networks, and service levels necessary to support applications. While there may be some variations, infrastructure typically includes servers, routers, switches, firewalls, information and communication networks, disk storage, middleware, and all of their associated terms (operating system, memory cache, raid level, etc.). The terms Infrastructure and Grid are used by some people as synonyms, while other people do not treat the two as exactly the same definition. For the purposes of this document we will treat them as separate concepts that are extremely similar.

5.2.1.2 **Grid**

The term "Grid" does not have a clear, universally accepted definition. The main body developing specifications in this area is the Open Grid Forum (OGF). Its scope is stated as being Applied Distributed Computing which clearly positions its relevance in IT infrastructure and virtualization. This note summarizes the state of standardization in this technology area. The focus is principally on standardization to support interoperability across different organizations. It does not consider standards aimed at allowing, for example, a single datacenter to be built using heterogeneous hardware and management systems from multiple vendors.

5.2.1.3 **Grid Middleware**

There are several standards-based Grid middleware systems currently available. Probably the best known is the Globus Toolkit which is very widely used, particularly in the USA. Much of the work in OGF has focused on this system and Globus includes implementations of a number of the OGF specifications. Other Grid middleware systems with significant user bases include UNICORE, Grid Resources for Industrial Applications (GRIA), gLite and Centralized Resources Over Wide Area Network (CROWN). These are still most widely used in scientific settings and typically require considerable expertise to set up and use effectively.

Commercial applications of Grid technologies in, for example, the finance, pharmaceutical, health, oil and energy sectors were originally focused on cost savings resulting from improved utilization of computing resources. Vendors addressing these markets typically specialized in particular sectors and developed proprietary Grid middleware solutions which could be deployed into commercial environments. Many have now diversified into other sectors but the result is that many Grids in commercial use are typically not standards based or easily interoperable with systems from other vendors.

5.2.1.4 **Grid Services**

The basic architectural standard of the OGF is the Open Grid Services Architecture (OGSA). This positions Grid Services as a specialization of Web Services (a Grid Service is essentially a Web Service associated with stateful resources). Therefore the Grid community adopts and contributes to more general Web Service standardization activities in OASIS, DMTF, W3C etc. As a result, the coherence and consistency of Grid standards is in a similar position to Web Services. There is a complex landscape where it is generally difficult to have confidence that specifications offer a complete, consistent set of technical solutions.

The NextGRID project (European Union [EU] funded collaborative) has recently finished. This project involved many prominent members of the Grid community in Europe, including strong connections to the OGF. This project aimed to define architecture for Next Generation Grids (i.e. standards-based Grids suitable for business). A major output from this project is a set of Generalized Specifications which profile specifications (from OGF and elsewhere) and provide guidance on how to use them (available from http://www.nextgrid.org). This project has made significant contributions to OGF over the last few years.

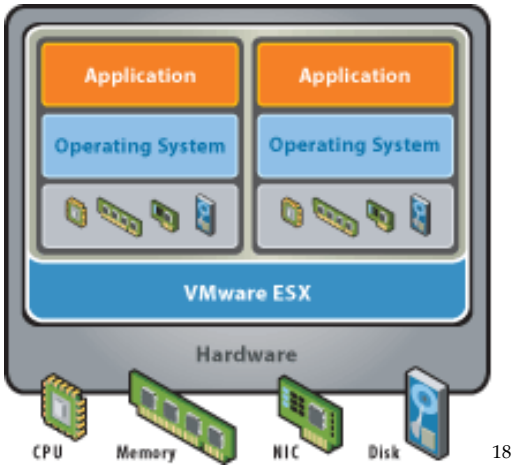
5.2.1.5 **Virtualization**

Virtualization is an umbrella term for enhancing some piece of the infrastructure's ability to do work.

5.2.1.5.1 Server Virtualization

The ability to run multiple logical instances on one physical machine and allocate resources dynamically to each environment can be found in server virtualization technology (such as the ones provided by VMware and XenSource). Flexible sizing, coupled with the ability to run multiple containers in parallel on a single physical server, maximizes the utilization of infrastructure. [17]

17 http://soa.sys-con.com/node/523458?page=1



18

Figure 5-6: Server Virtualization Example

In essence, server virtualization allows a provider to run several logical servers on one physical server. The physical server typically has more resources at it's disposal than a "standard" server. One physical server, even one large enough to support many logical servers, is typically cheaper than buying many logical servers. When several large physical servers are combined into an infrastructure, the number of logical servers able to be supported increases significantly. So the cost to support many logical servers becomes more cost effective from a server cost and utility cost (electricity, cooling, etc.) perspective.

Virtualization can also be implemented so that a logical server can be moved from one physical server to another, in some cases automatically at signs of server distress. This gives the infrastructure more flexibility and redundancy than individual servers would have.

Because of the cost savings, increased flexibility, and increased redundancy, server virtualization has attracted a lot of interest from Infrastructure groups.

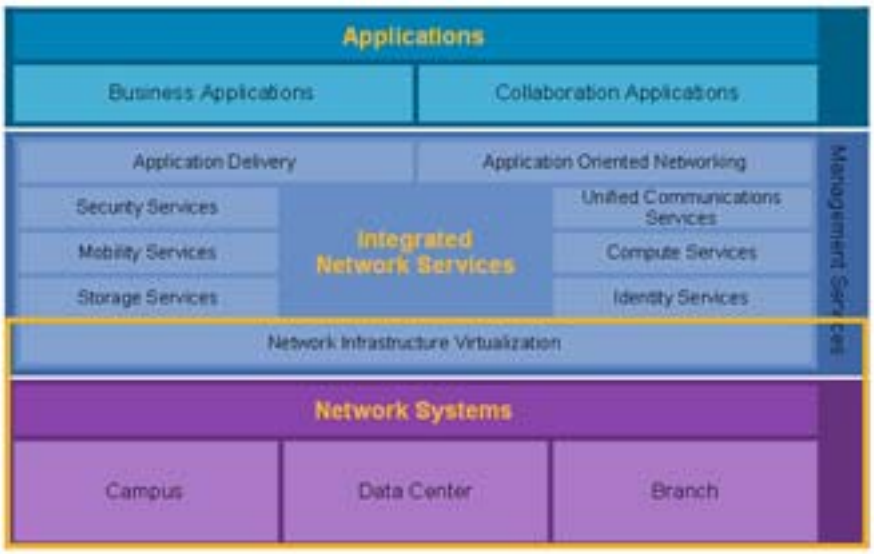
5.2.1.5.2 Network Virtualization

The second kind of virtualization, which allows for multiple distributed replicas of an application module to appear as one to the external world, can be found in network application switches. To get the most out of SOA, the application should run on an adaptive, programmable, and application-aware network that connects the user to the different services and the application modules to each other. Since the service "containers" change in size and physical location constantly (based on demand,

18 http://virtualizationworks.com/VMware-ESX-Server.asp

physical constraints, and geographical cost factors), the network should be able to adapt, so that all changes are seamless to the end user as well as to additional application modules up or down the logical stream.

The network must be able to make the necessary decisions to assure that there are no interruptions to business services due to these internal or external changes. Today, this kind of adaptive network is supported by application switches (such as Citrix's NetScaler). These switches provide an API that can change policies on-the-fly as well as view and change parameters in application payloads (beyond the existing protocol fields handled by regular switches).[19]



[20]

Figure 5-7: Network Virtualization Example

5.2.1.5.3 Storage Virtualization

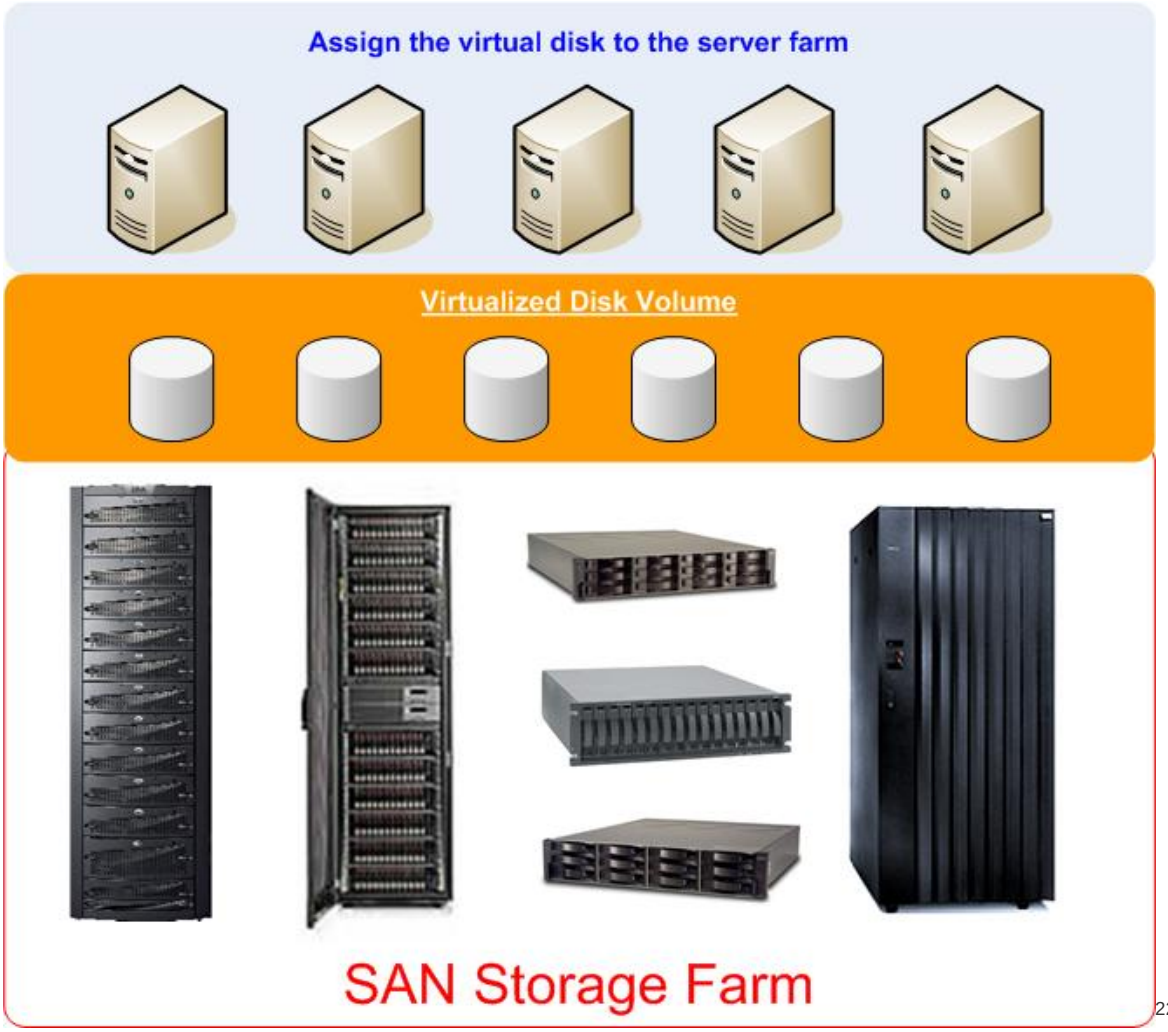
Treating storage as a single logical entity without regard to the hierarchy of physical media that may be involved or that may change. [21]

In essence, storage virtualization allows a provider to configure disk storage that may run across one or more "back office" hard disk devices (typically a Storage Area Network (SAN) device).

19 http://soa.sys-con.com/node/523458?page=1

20 http://www.cisco.com/en/US/netsol/ns658/networking_solutions_package.html

21 http://www.techweb.com/encyclopedia/defineterm.jhtml?term=storagevirtualization



[22]

Figure 5-8: SAN Storage Farm Example

5.2.2 Software Virtualization or Software As A Service (SaaS)

Software as a service is a phrase used to describe a delivery model for a software application in which a software supplier hosts and operates the application. Customers access the service via a network (often the Internet). This model is distinct from the more traditional usage of software in which customers host and operate applications on their own computers. A good example of this difference is that between Microsoft Office applications and those from Google Docs. With

[22] http://www.microware.com.hk/cms/spaw/images/newsletter/Storage%20Virtualization%20Diagram.jpg

the former, a user installs the Office suite on their PC and then uses each application directly as they require. With the latter, each application is accessed by connection into Google's computing estate over the Internet.

Early forms of Software as a Service (SaaS) suffered from difficulties associated with running applications remotely, namely a) only being available when connectivity is available; b) poor network quality of service leading to poor application performance; and c) poor network security compromising customer data. With the higher availability of secure broadband networks and the ubiquity of access methods such as cellular radio and Wi-Fi, SaaS is now a well-established model for application services with customer relationship management (CRM) and e-mail leading the way. Ovum estimates the worldwide SaaS CRM market to be worth USD 1.3 billion by 2010.

SaaS can be offered with a user interface (in which the user interacts with the service via an interface offered by the supplier, often through an Internet browser) or a developer interface with which customers create their own user interface which in turn connects to the SaaS application via the Internet. This allows for greater customization of the way in which the application operates, however a customer created interface introduces a new set of security challenges for the Service Oriented Network. SaaS vendors deploy load-balanced and scalable computing architectures which can be tailored to match demand from the system and in which service upgrades can be introduced without lengthy downtimes.

Business processes are increasingly being re-engineered to support new agile ways of working and companies are out-sourcing many of these processes to low cost providers, often in emerging markets, in an effort to concentrate on core business opportunities. The provision of IT services also falls into this category and SaaS provides companies with an opportunity to dispense with the need for large-scale computing infrastructure and expertise that is required to install, run and maintain software 'in house'.

One of the key features of SaaS is the use of a subscription or usage-based pricing model for the service. That is, users pay only when they are using the service provided by the application. This differs markedly from the more traditional licensing model in which users pay a licence fee (and often a regular maintenance fee) to the supplier to have the software installed on their own computers. Such licenses are often offered on a per-seat basis, i.e. the more installations the higher the cost. The licensing fees will often be the same whether the installations are used or not. SaaS, therefore, substitutes cost-of-ownership with cost-of-usage.

For users, SaaS offers a low-cost alternative to license-based applications, particularly where usage patterns are expected to be low. As a result, software suppliers tend to have lower unit returns from SaaS than from traditional licenses. However, by offering a lower cost alternative, software suppliers hope to gain higher market shares. Software suppliers tend, therefore to adopt SaaS where the application has a high general applicability such as e-mail or CRM. However, SaaS also offers a cost-effective channel to market for 'long tail' or other niche applications due to the low market entry costs. One of the more well-known SaaS products to

have emerged in recent years is that provided by www.salesforce.com in which CRM services can be accessed via the Internet.

5.2.2.1 **Standardization in Virtualisation**

Virtualization is largely incidental to the use of IT infrastructure resources in service oriented systems. Use of virtualization technologies can give additional flexibility to the provider of IT infrastructure services but is certainly not essential unless, for example, the services offered are explicitly to host virtual machines. Currently there are a number of incompatible VM formats (from VMware, Xen, Microsoft, Sun, and Amazon for example). There is a proposal within the DMTF to develop OVMF (Open Virtual Machine Format), which should allow a single virtualization platform to run virtual machines in any of the existing formats.

5.2.3 Why is IT Infrastructure important to a Service Oriented Network?

SOA is defined as a collection of services that communicate with each other via simple data transmission methods or more complex activity coordination.

A service is defined as a function that is well-defined, self-contained, and does not depend on the context or state of other services.[23]

In the early stages of a service oriented network, a service provider may be able to run SOA applications on current infrastructure models. However the growth of SOA services tends to be exponential. "In an SOA environment, it's a challenge to allocate resources or govern usage and service levels effectively. The cost of scaling and manual support operations rapidly begins to grow in an exponential pattern that makes current operations models unfeasible." [24]

In order to provide "well-defined, self-contained, independent services" and meet the highly unpredictable need for resources, a complete SOA Architecture model needs to use Server, Network and Storage Virtualization along with Service Level Automation. The OGF, The Open Group and others define this infrastructure layer as Service Oriented Infrastructure (SOI).

5.2.3.1 **What is SOI?**

[23] http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

[24] http://soa.sys-con.com/node/523458

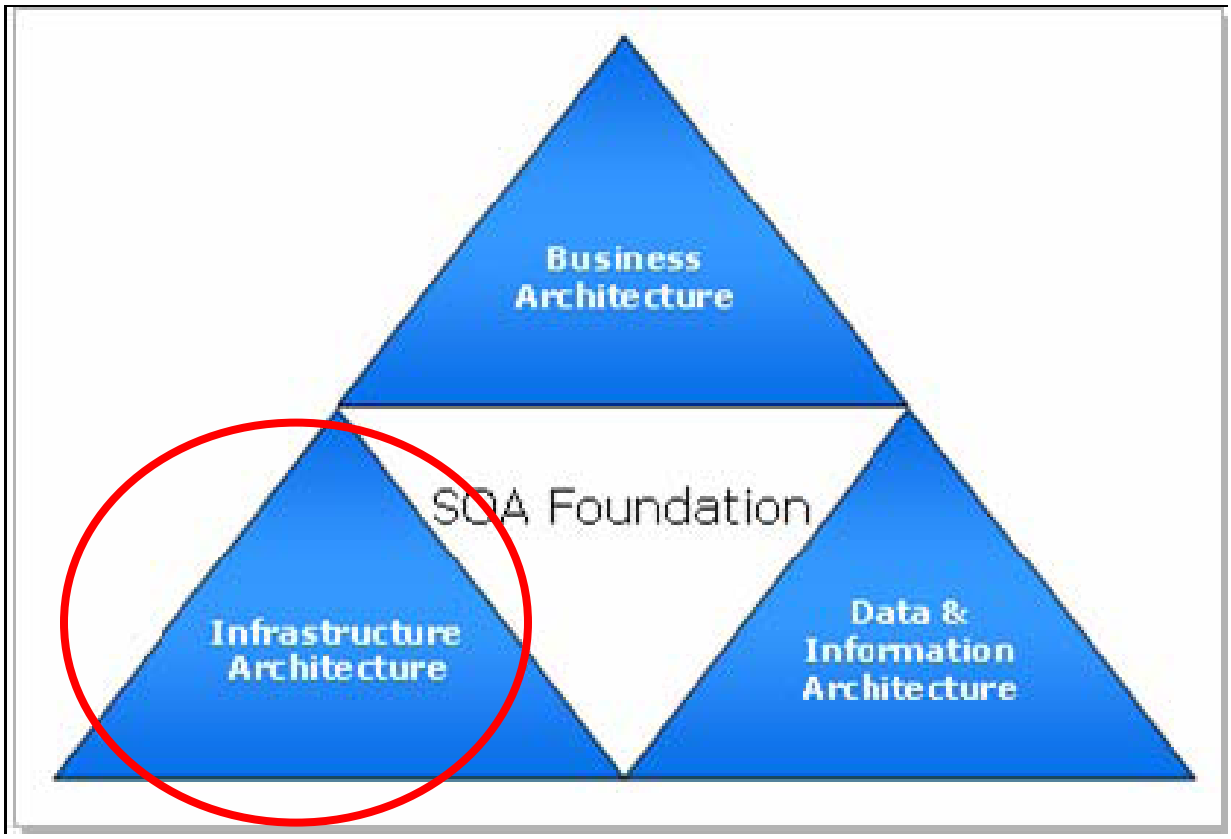


Figure 5-9: Service Oriented Architecture

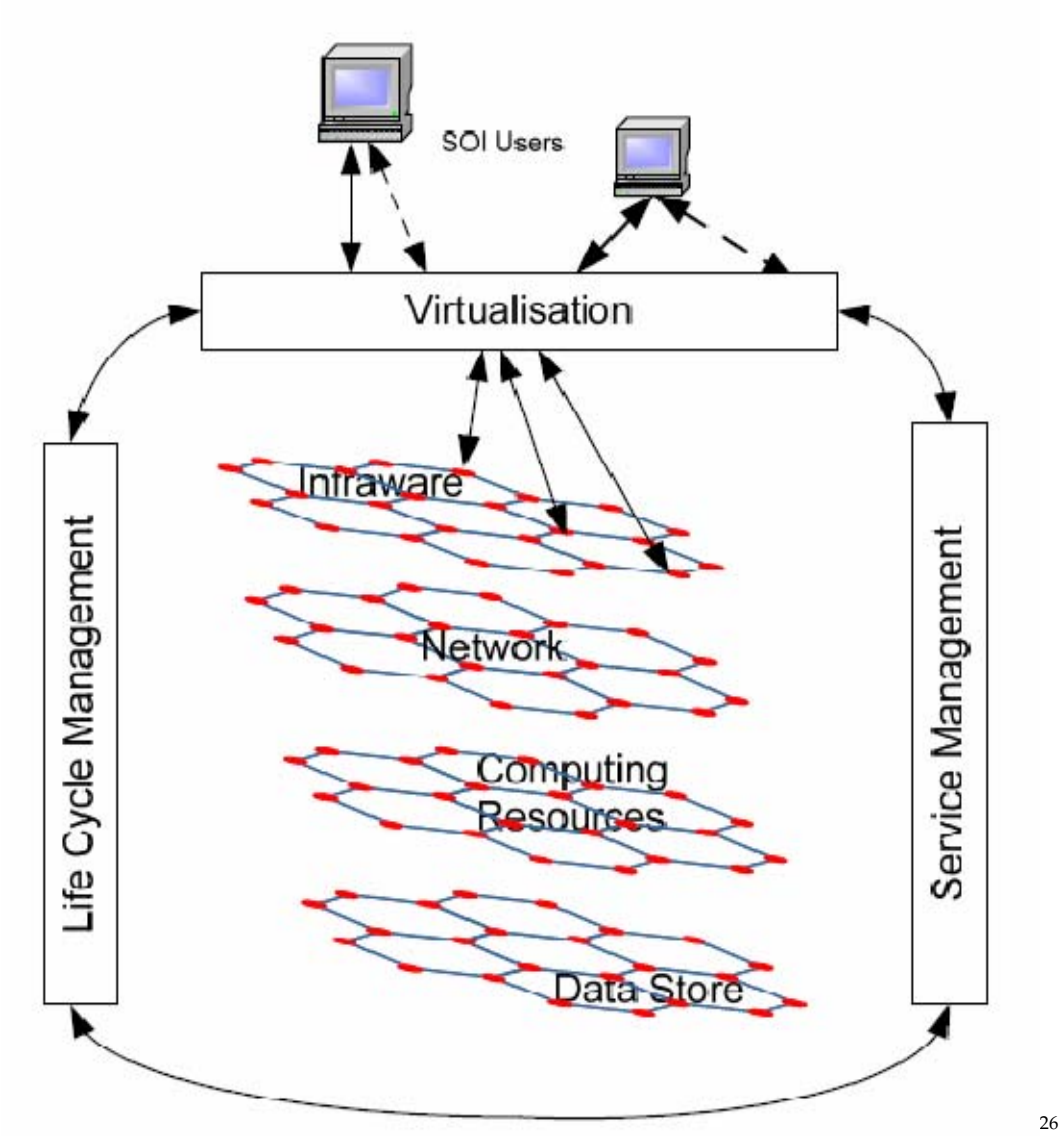
"SOI is related to *Service Oriented Architecture* (SOA) which is most commonly used to refer to the use of this principle (meaning SOA) in Application Architecture. A Service-Oriented Infrastructure forms an appropriate foundation for a service-oriented Application Architecture, and can be regarded as a natural part of a service-oriented Enterprise Architecture. " [25]

The three building blocks for an SOA foundation are the Business layer (or Application layer), the Data & Information layer, and the Infrastructure layer. SOI directly relates to the Infrastructure layer.

A lot of discussion and press about SOA has been about applications which can lead to some confusion, because so many times when people talk about SOA they are referring to the Business layer of SOA. A complete Service Oriented Enterprise Architecture includes the Business, Infrastructure and Data layers.

[25] http://www.opengroup.org/projects/soa-soi/uploads/40/14171/OG-SOI-Charter-v1-1.pdf

When implemented SOI takes advantage of virtualization and service level automation to create a real time adaptive environment that meets customer's ever changing expectations and helps manage the service provider's cost of service delivery.



[26]

Figure 5-10: Generic SOI Elements

SOI and Infrastructure layer tools are not mandatory to provide services in a Service Oriented Network. Most Legacy services will not fit into an SOI model very readily. So, these Legacy services could be addressed at the application layer only. However, by making their Infrastructure fit an SOI model, it does allow a provider to provide more flexible, varied, and in depth services in their Service Oriented Network.

[26] http://www.opengroup.org/projects/soa-soi/uploads/40/14171/OG-SOI-Charter-v1-1.pdf

5.2.3.2 *Standardization in SOI*

Service oriented infrastructure is a relatively new term which refers to the exposure of functionality traditionally closely associated with Information and Communication Technology (ICT) infrastructure as a set of services. This is not a wholly new area - there has been considerable work in standardizing service oriented systems and in standardizing various aspects of ICT infrastructure and its management. Standardization relevant to SOI is complex and involves many different bodies with different approaches, scope and style. There is certainly a lot of activity but it is unlikely to be coherent or consistent.

5.2.4 Security and Digital Rights Management Considerations

In a true SOI environment security can become an extremely important issue. If a provider's SOI is trading or shifting computing, storage and networking resources back and forth, it will be paramount for the entire infrastructure to know what security levels/Digital Rights exist and have been approved for a user. If a customer's "infrastructure" is compatible with the provider's SOI, and the provider allows the customer to use some of their own "infrastructure"; there needs to be an existing method or standard on how the Security and Digital Rights will be managed and tracked. Therefore any SOI plan for SON should ensure it has considered how to handle security and digital rights management.

5.2.5 Examples of services using grid computing, infrastructure, and/or virtualization

5.2.5.1 **Example 1**

A TV customer flipping through their IP TV menu decides to take the provider up on their new Personal Video Recorder (PVR) offer. As part of the menu selections, the customer chooses to purchase a virtual PVR with a standard amount of disk storage, but pay a little more for a premium service with a higher degree of reliability in regard to latency (this customer can't stand lag of any kind). After the provider configures the virtual PVR and disk storage and makes it available to the customer, the customer logs on and decides to go augment the disk space on his PVR by using some disk storage in his own house. He accomplishes this by adding a virtual disk on the PVR which points to his disk storage at home; however, he does not do anything special with latency because he isn't worried about the latency in his own network and disk storage.

5.2.5.2 **Example 2**

A movie producer orders a movie making package from the provider. The provider provides the producer with a virtual server (including the latest movie making SaaS application), virtual storage for the server, and recording devices with broadband mobile cards. As the movie is being recorded the recording devices will attempt to connect to the virtual system to store what is being recorded. If the broadband network is unavailable or there are latency issues, the recording device will record the movie onto it's local storage, warning the operator when the device is getting "full". The

recording devices will continue to poll the system so it can download recordings when the network is available or latency is back to acceptable levels. As an option the movie producer chose to purchase the movie making application (which was loaded on the virtual server) with a standard amount of CPU, but the availability for it to increase on demand for a fee. She knows there will be times when her editors and Common Gateway Interface (CGI) people will log onto the system and push the CPU limits above standard, but with appropriate planning those times should be minimal.

In this example the provider was able to trade off disk storage, computing power and network resources based on various conditions. So if these capabilities are in "the tool box" a provider will be able to provide more flexible and varying services.

5.2.6 *IT Infrastructure Use Cases*

The following use case expands on the Ad Funded Navigation use case shown in Section 2.

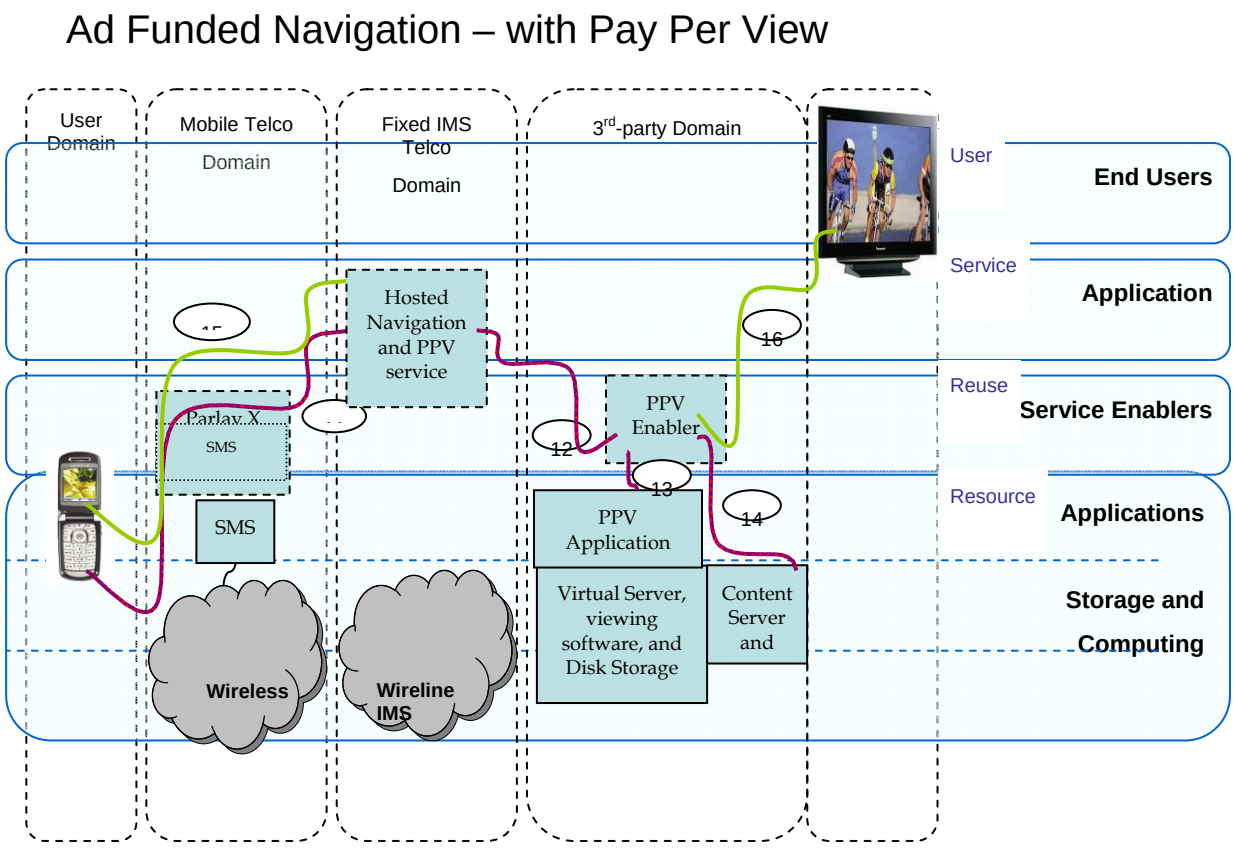


Figure 5-11: Ad- Funded Navigation

11. The Mobile User clicks to accept the Pay Per View advertisement and invokes a Parlay-X Pay Per View service.

- Charging event (Pay Per View Ap pays Nav Ap and Ad Ap)

12. The Navigation application invokes the 3rd party web service to initiate a Pay Per View event.
13. Pay Per View application initiates the creation of a Virtual Server and Disk Storage to host the Pay Per View event which includes the loading of appropriate software to deliver the service.
14. Pay Per View application also prepares the content server to feed content to the Virtual Server when authorized.
15. Confirmation of success is sent back to the mobile user along with security information (pin) to start the event.
16. After the customer arrives at the restaurant and is seated at the table, the customer enters the appropriate pin/credit card # information on the screen. The display screen invokes the 3rd Party Web Service to check authorization and if approved, start streaming the content to the screen.
17. The Pay Per View event delivers charging events to billing center.

5.2.7 Standards Assessment of IT Infrastructure

5.2.7.1 **Overall Assessment**

Standards for Grid, Infrastructure, Virtualization and SOI are so entwined it can be difficult to separate the individual pieces. However, it is safe to say, if you have SOI standards, the SOI standards will include Grid, Infrastructure and Virtualization standards.

There is a significant amount of discussion about SOI standards in bodies like Open Grid Forum and The Open Group. Certain elements of the SOI have standards or have standards bodies working on them. The Storage Networking Industry Association (SNIA) formed recently to develop and promote standards in the Storage industry. TM Forum has some working groups working on projects involving infrastructure solutions or challenges. In late 2007, DMTF (www.dmtf.org) released a data model to support data necessary to support SOI and virtualization. DMTF continues to release updates to the model. In 2008, NextGrid (www.nextgrid.org) released their final report. Their project was "to develop an architecture for next generation Grids to enable their widespread use by research and industry".[27] European Telecommunications Standars Institute (ETSI) has formed a TC Grid committee which aims to promote convergence between IT Resources and Networking.

Current standardization efforts are focused on academic and research communities. Projects are currently underway that assess the right models into which any standard can be adapted. Given this stage in the lifecycle, it looks as if the industry wont have standards that could be used by operators and vendors until the research work is complete – not likely for a number of years yet. Until then, operators wishing to include

[27] http://www.nextgrid.org/download/publications/NextGRID_Final_Report.pdf

SOI, Grid/Infrastructure, and virtualisation in their SON offerings are restricted to proprietary deployments.

SOI standards are not mandatory to provide service oriented applications, but the service oriented applications without SOI will be limited to only what software and proprietary systems can provide. A provider without an SOI in place will negatively impact the types and flexibility of services they are able to provide.

5.2.7.2 **Recommendation**

Recommendation removed for external transmittal.

5.3 OSS/BSS

The **Operational Support Systems (OSS)** are usually referred to as the fulfillment and assurance systems that are used by carrier service providers to operate, administer, plan, and maintain their infrastructures. They form the operational systems and process workflows of day-to-day strategic operations for the service providers. **Business Support Systems (BSS),** complementary to the OSS, are the customer-facing systems used by service providers to win, add, and maintain customers. They form the backbone of the order-to-cash business process and functions for the service provider to customer relationship. Billing, customer care, sales, marketing, and related Enterprise Resource Planning (ERP) systems are usually classified as BSS systems.

Both OSS and BSS are terms more commonly used by carrier-based service providers. The Internet-based service providers such as Yahoo, Google etc., may not use these terms however they have similar functions/systems internally to carry out the enterprise operations, these new entrants of service providers are facing similar challenges as the traditional service providers in managing converged voice, data and media services.

In the face of Internet, Media, and Telecommunications industries convergence, for traditional ICT service providers a critical requirement is to create a new market ecosystem formed by this convergence, referred to herein as the "ICT hyper-sector". This new market brings together the Telco and non-Telco worlds, leveraging the ubiquity of one and the innovative pace of the other. This new ecosystem must meet the business goals of all of its members through facilitated cooperation in the entire service lifecycle management such as service creation, fulfillment, management and billing.

Non-traditional ICT service providers face the problem of securing a large and stable base of paying customers. Many of these providers generate revenue from advertisement or small fees on transactions. To increase revenue they also need to diversify their offerings and take advantage of the "infinite" shelf that the Internet offers

(e.g.: Amazon.com). Web 2.0 is a way to fill this shelf but it lacks the means to manage and commercialize the results.

Providers of SaaS are also on the rise. They too aim at securing a large and stable base of paying customers to create revenue. Their goal is to constantly deliver on customer expectations meaning these customers can run their own business based on such guarantees.

Also, as services are becoming more complex, particularly as service providers look to repackage services into bundles and "mash" them together into new products. Ultimately, customers will drive product composition, creating bundles of services personalized to their unique requirements. Furthermore, SPs want to present customers with an intelligent catalog of services that understands the rules for composing services together into products, given the customer's network connection(s), pricing rules, and other relevant information, such as the customer's status, affiliations, or credit history. Once the customer has composed a suitable product, it will become a new service that must be provisioned, charged for, and assured – and customers increasingly expect that such services will be delivered on time, available when wanted, and competitively priced. These expectations, as well as the new model of product composition, are driving new operational requirements.

The management processes of all providers must evolve as the industry evolves, as well as the models that guide and automate these processes. The emerging management models must, in particular, deal with the following key requirements:[28]

- **Mass-market scale of consumer driven services**, demands more service automation, since manual processes for per-customer support and customization makes any business operations an unacceptable business cost and risk.
- **Accelerated time-to-market,** to address the evolving consumer requirements and to reflect the velocity of change that has made the Internet successful. This must include the ability to introduce, change, customize, repurpose, and retire services and service components quickly.
- **Increased use of hosted services**, both internal and external, to create new services and thereby to reduce time to market, improve customer and partner satisfaction, and reduce churn, relative to service-specific development. Service offerings need to be more responsive to the market changes and to reflect the growth of ICT companies who use the communication infrastructure provided by others rather than create their own.
- **Support both current service providers and new entrants** to compete at the new marketplace as efficiently as possible (e.g. cost wise) at "internet speed". i.e. Supporting

[28] Excerpts from TMF TR139

different business models such as having a service broker role or leaner infrastructure, such as hosted infrastructure.

- **Increased partnership among all forms of service providers** to enable the marketplace to create new services or experiences sold to the customers rather than a market where multiple providers simply field competitive offerings. This alters basic conceptions of customer and resource ownership and extends the dimensions of all management processes across organizational boundaries.
- **Support multi-dimensional revenue streams from consumers and providers,** as componentized services and features can be mixed-and-matched from many different providers and sold through many different channels to many different end users, the service providers must be able to ensure the ability to capitalize its revenue streams from all opportunities. Support payments for content acquisition/use, the use of resources outside the Service Provider's domain such as SEs, revenue characteristics of/options for services being created and deployed through a SDP, and financial aspects of service syndication. Support is needed across a broader scope of services and features and customization, and across a broader set of revenue-generation models (e.g. advertising/dynamic promotions/tie-ins as well as usage-based or flat rate). Additionally, many service providers are looking into offering their current OSS/BSS infrastructure as reusable components that can be bundled into other service components as part of revenue stream. Support may require interaction with User Profile, Subscriber Profile and Service Policy entities.

As service providers transition their operations to provide new classes of value-added services, they have the opportunity to transition to a new real-time service management model which leverages existing OSS infrastructure, but greatly simplifies and speeds the creation and delivery of new services. Taking advantage of these capabilities, however, will require several incremental steps to enhance operations with the aim of enabling faster and less costly service delivery; centralized service creation, subscriber service management and real-time session and transaction-based service capabilities.

5.3.1.1 **Recommendation/Identified Gaps**

Recommendation removed for external transmittal.

5.3.1.2 *Moving Towards Service Oriented Architecture and Agile Service Delivery*

All of the above requirements call for a loosely coupled OSS/BSS architecture to provide the agility required for the full service lifecycle management, i.e. from service design, creation, deployment, operation and to retirement. Many service providers and OSS/BSS vendors have moved towards the Service Oriented Architecture to satisfy these needs.

While adopting an SOA design philosophy across the OSS/BSS stakeholders would provide a more flexible architecture where operators will be able to support many different business models and roles that each Service Providers would like to play, the

current SOA standards and methodologies developed by the IT mainstream will need to be enhanced in order to support the complexity of real-time communication services and domain specific standards, such as:

- **A federated information architecture which serves as the ontology definition for the communication services domain.** As the new services model introduce a great emphasis on creating innovative services that are rapidly deployed and tied to users need. Operators and Service Providers are therefore facing important challenges to manage the overloaded data of the user preferences, the customized services and the whole service delivery chain. This is mainly due to the lack of standardization and common vocabulary, which strongly hinder information exchange and communication between information systems. Ontologies, which capture the semantics of information from various sources and giving them a concise, uniform and declarative description is needed. TM Forum has proposed SID as such Information Architecture to be included in the ITU-T NGN Focus Group studies back in 2006. Although one may argue SID is not complete to be used as an ontology model, it remains to be the most comprehensive information model that is available in the industry which is implementation agnostic and can be extended to support various technology domains such as VoIP, Web 2.0 services etc.

- **Service contract and enhanced SOA protocols** (e.g. web services) that goes beyond the traditional WSDL/web services to include attributes required for long running service transactions, asynchronous notification and non-functional elements such as reliability, fault and QoS. There are several standards bodies working in this area, including the newly formed OASIS Telecom Service Sector to address the protocol issues as well as work in 3GPP Integration Reference Point (IRP), the TM Forum NGOSS contract metamodel definition as well OSS/J, Internet Protocol Detail Record (IPDR), TMF/Multi-Technology Operations System Interface (MTOSI), IPsphere working at the service template and interfaces specification areas.

- **OSS/BSS service components,** while the term "service enablers" are most often used to refer to the capabilities or the capabilities as part of an application delivered to the end customer, in the realm of composite services, it is also envisioned that OSS/BSS components can be packaged as enablers to be bundled into the application such as charging, fault reporting etc. The TM Forum Telecom Application Map (TAM) effort was initially developed to provide an application view to support the Enhanced Telecom Operations Map (eTOM™) business processes and required information (SID) exchanges. The use of TAM in the context of identifying touch points in the service delivery value-chain and required OSS/BSS service enabler definitions are currently being worked in the TM Forum SDF team.

5.4 Security and Trust Models

5.4.1 Overview

A key concept in SOA is that re-usable service components can be developed by independent third parties organizations, and that these can be invoked by carriers. It seems fairly obvious that this will require a robust security framework, and that certificates could be an important part of that security model.

The ability to coordinate content ownership, and control in a multi ecosystem environment also includes consideration of requirements for certificate authorities and impact on existing trust models, particularly when considering the concept of applications as content and the exchange of service enabler information. By implication, there is likely to be a need for a Certificate Authority to manage these certificates.

5.4.2 Standards

OASIS Web Services Secure Exchange (WS-SX) TC[29] is currently addressing a number of issues in this space, and certificates do in fact play a key role in their documents. In particular, the WS-Trust document[30] provides a full trust model and a complete description of the use of certificates in that model.

In addition, the newly formed OASIS Telecom MS also plans to address this topic, and to identify any additional requirements that might apply to telecom applications. It appears likely that this will be part of the use-cases and requirements in the initial work performed by the Telecom MS.

5.4.3 Recommendation

Recommendation removed for external transmittal.

6 POLICY IN THE SERVICE ORIENTED NETWORK

This section provides an analysis of the broad area of policy in the SON. Additional information, including usecases, may be found in Appendix D.

[29] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx

[30] http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

6.1 A Definition and Context

A policy, in the context of a telecommunication network, represents a set of rules applied to two or more communicating end-points. These rules represent conditions and actions that either "enhance" or "restrict" the communication and the functionality provided between these end-points.

A policy is applied to a communicating end-point by a "Policy Enforcement Point".[31] The decision as to what action to apply to the communication between end-points is provided by a "Policy Decision Point". The "policy condition" supports processing of requests for *authorization* originating from policy enforcement point to a policy decision point. The policy enforcement point will perform (enforce the decision authorized by the Policy Decision Point (PDP)) the appropriate action upon receipt of the authorization response originating from the policy decision point. Policies can be defined and enforced by the communicating end-points or by the entity controlling the communication between the end-points.

Subscriber Policies are communication rules that are defined for the purpose of controlling how the subscriber communicates with other endpoints and for controlling what information is allowed to be exchanged. Examples of subscriber policies are:

- Functionality offered based on subscriber availability status
- Functionality offered based on subscriber connectivity status
- Functionality offered based on terminal capabilities
- Functionality offered based on Time of the Day/Date
- Functionality offered based on subscriber location
- Functionality offered based on subscriber's permissions and preferences; e.g. privacy, called/calling party
- Functionality offered based on subscriber's identity attributes
- Functionality offered based on subscriber's availability of funds

Network Policies are communication rules that are defined by the network operator that control the communication resources used for communication between end-points. Examples of network policies include:

- Bandwidth Management
- Access Control
- Latency Control
- Priority Services

6.1.1 Standards and Policy

[31] [IETF PEP-PDP model-RFC 2753]

There are a large number of standards that cover policy as used in the SON Framework including:

- ITU-T Recommendation X.812 (1995) | International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 10181-3:1996,Information technology - Open Systems Interconnection – Security frameworks for open systems: Access control framework
- ITU-T RECOMMENDATION X.749:1997 | ISO/IEC 10164-19:1998,Information technology - Open Systems Interconnection – Systems Management: Management domain and management policy management function
- ISO/IEC 21000-5:2004, Information technology – Multimedia framework (Motion Picture Experts Group (MPEG) 21) -- Part 5: Rights Expression Language, ISO/IEC 21000-6:2004, Information technology – Multimedia framework (MPEG-21) -- Part 6: Rights Data Dictionary
- ANSI/International Committee for Information Technology Standards (INCITS) 359-2004: Information Technology -- Role Based Access Control (RBAC)
- RFC2622: Routing Policy Specification Language (RPSL)
- RFC 2753: Policy Based Admission Control
- RFC2748: The COPS (Common Open Policy Service) Protocol
- RFC3060: Policy Core Information Model (PCIM)
- RFC4011: Policy Based Management MIB
- RFC 4745: Common Policy: A Document Format for Expressing Privacy Preferences
- W3C Recommendations (standards)
  - Platform for Internet Content Selection (PICS)
  - Platform for Privacy Preferences (P3P)
- WG Note: Web Services Architecture (WSA)
- Workshop: Constraints and Capabilities for Web Services
- Web Services Policy 1.2 – Framework (WSPolicy)
- OASIS WS-Security Policy
- OASIS Extensible Access Control Markup Language
- (XACML)
- OMA - Policy Evaluation Enforcement and Management Enabler (PEEM)

6.1.2 OMA Policy Evaluation Enforcement Management Enabler

The Open Mobile Alliance defines several policy based "service enablers. Work done in OMA regarding policy includes:

- PEEM
- Global Permission Management Enabler (GPM)
- General Service Subscription Management (GSSM)
- Categorization Based Content Screening (CBCS)
- XML Document Management (XDM)

The work done in OMA PEEM in the policy space can be leveraged as a model and framework for detailed policy discussions. OMA PEEM is chosen as a model for policy discussions because of the strong reliance by OMA on "Service Enablers" and the fact

that ATIS SON-FG has also recognized "Service Enablers" as one of the key building blocks for Service Oriented Networks.

In OMA policies are enforced by enablers such as the PEEM. The PEEM evaluates, enforces and manages policies. Policies are used in effect to "compose" multiple services (e.g. charging services, location services, privacy/security services, Interaction with a third party service provider) into a more complex service. The two main patterns supported in PEEM are the "***proxy***" and the ***callable***" pattern.

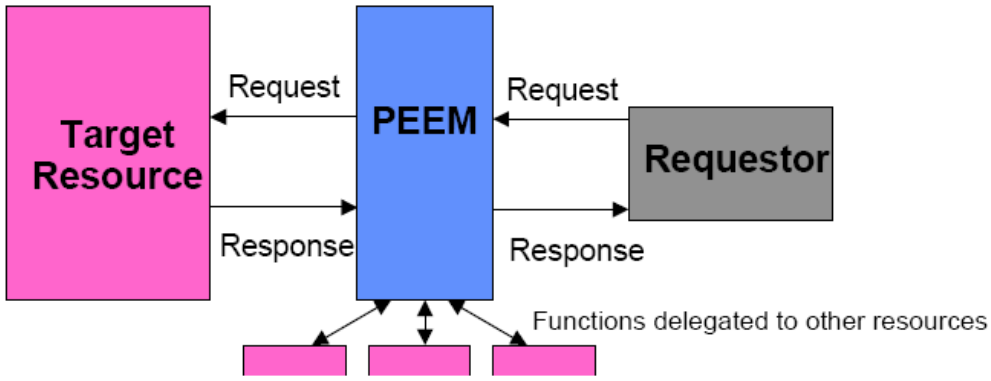


Figure 6-1: OMA PEEM as Proxy Usage Pattern[32]

An example of the OMA Proxy pattern in use today is available in Web Proxies that support additional policy functions (functions delegated to other resources) like content adaptation or content screening via use of interfaces such as Internet Content Adaptation Protocol (ICAP) or the Open Pluggable Edge Services (OPES) protocol. In this model a web proxy (PEEM) would delegate to specialized servers the task of, say screening the HTTP content being sent by the requestor to a given target resource (Web Server).

[32]OMA defines a new advanced policy management standard for mobile network. It has web-based interfaces and integrates and unifies different concepts of policy model such as the concept of the IETF and the WSS technical Committee at OASIS. The OSE enriches the policy model with the notion of assertion tokens for authentication and authorization and also provide a central point to manage workflow, delegation, and service composition. Whenever service or application needs delegation, orchestration or composition that must be expressed as policy and the PEEM is responsible to enforce this delegation

[32] Source "OMA Policy Evaluation, Enforcement and Management Requirements"

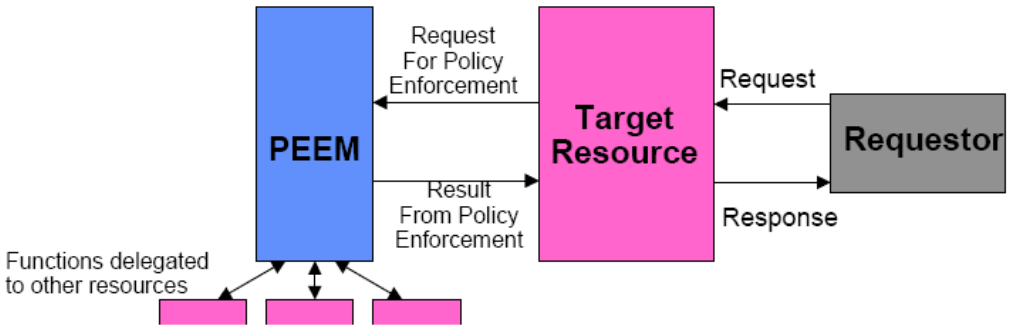


Figure 6-2: OMA PEEM as Callable Usage Pattern32

An example of the OMA callable pattern in use today is available in the IMS online charging architecture where network elements (Target Resource) request permission to provide a service, requested by a subscriber (requestor), based on availability of funds from an Online Charging System (PEEM).

6.2 Policy: A service enabler, enhancing user experience

Proper application of policies can both restrict and enhance the user experience as a subscriber is utilizing a service or application. Restricting access to location information for family members is an example of how policy controls/restricts access to personal subscriber information. Policy as a "service enabler" for service oriented networks is an important concept in the area of "personalization". In order to understand the richness of the user experience possible when policies are applied, consider *Figure* 6-3 below. This shows how several service enablers (Multimedia Service Enablers) can play a "policy" role in the lifecycle of multimedia content (ring tones, wall paper, songs, and movies). The relationships between the multimedia content use cases and the service enablers, depicted in the "Personalization Domain", are realized based on the enforcement of policies.

Policy becomes a service *choreographer* (see the "Policy" relationship to other in 6-3 below), coordinating requests that need access to the operator or subscriber resources (subscriber profile, privacy information, location, etc). When policy enforcement involves delegation to multiple enablers, it actually amounts to performing Service Oriented Architecture (SOA) composition to create *new functions* (i.e. SOA orchestration). Policy Enforcement is viewed in this context as an orchestrator.

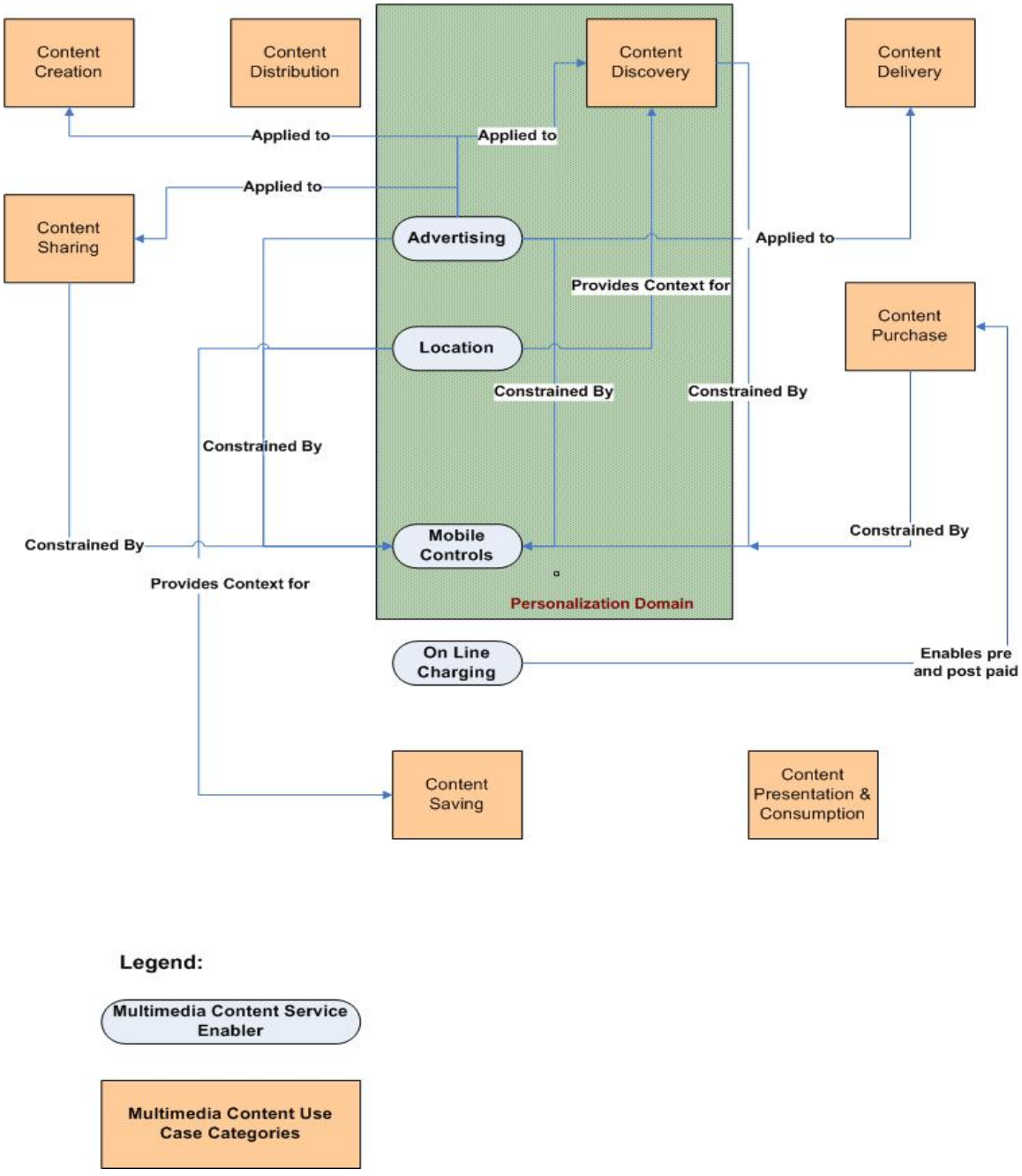


Figure 6-3: Service Enablers and policy create a personalize user experience

6.3 Policy Gap Analysis Summary

This section, and supporting text in Appendix D defines the different phases of the "*policy life cycle*":

- Policy Detection Phase
- Policy Context Phase
- Policy Decision Phase
- Policy Enforcement Phase

In the area of "Policy Detection" the Gap Analysis describes how the policy "proxy" and "callable" policy patterns are not always sufficient to be able to determine when specific policy rules need to be enforced. In standards and typical operator implementations, triggering of policy decisions, is based on the "proxy" and "callable" policy patterns. These solutions are implemented either in the access network (using the proxy policy pattern) or at the target server (using the callable policy pattern). The section describes the challenges operators face with these approaches **and proposes an alternative "*hybrid client*" based policy pattern which helps overcome the previously described challenges.**

In the area of "Policy Context" the Gap Analysis describes how policy rule engines need to receive information (such as subscriber availability, location, desired group of subscribers allowed or disallowed for communication, etc) from the network as input into the policy decision process. The information needed to make policy decisions originates from multiple policy information points (location server, presence server, parental control server, etc) within the network. The Gap Analysis section describes the challenges with standards trying to address this problem. Solutions to this problem are either not widely adopted or too specialized around a specific service. ***Instead, an alternative solution is proposed that leverages presence (a core enabler in IMS) as a mechanism to support collection of policy decision information from policy information points dispersed across an operator network.***

In the area of the "Policy Decision" phase, the Gap Analysis section identifies the area of policy "delegation" across multiple service enablers as requiring further study.

Finally, in the area of the "Policy Enforcement" phase, the Gap Analysis section describes the challenges of supporting the proxy policy pattern in the network.

7 BUSINESS MODELS AND PROCESSES

7.1 Introduction

In the past sections we have mainly described SONs and their standardization requirements from a hands-on, operational perspective: What needs to be in place so that applications and services within the SON paradigm work? In this section we will discuss changes and challenges from the internal and external business point of view:

- **Internal:** What are the requirements on SON and their applications and services so that CIOs and CTOs will implement and execute the SON vision – migrating existing applications and developing new applications under the new paradigm? One

example of an obvious hurdle would be a new technology that requires a completely different way customer information is stored.

- **External:** What are the requirements on SON and their applications and services so that they can be sold to customers? One example of an obvious hurdle would be new services that work great but can't be integrated into the customer's billing system.

There are three main questions that any migration or use of SON application and service implementations needs to answer:

1. What will that mean for me from a business perspective?
2. What will it do for my business, what is the business value?
3. What is the impact on my running operations, tactical planning, and strategic planning?

While this sounds more like a simple task of extracting Key Performance Indicators from services and their orchestration / integration, the devil is in the detail. We will first explain high level business requirements that are mainly brought upon us through the virtualization of "things", explore the resulting transformational business impact, and then derive the transformational impact on SONs.

7.2 Business Models

Building on the understanding of the elements and interactions within the SON framework, we turn our attention to the business models which may be represented in and supported by the SON framework. Business models within the industry are constantly evolving driven by among other things changes in market dynamics, technological advancements, competition, and customer preferences. While we can't hope to cover all potential business models representative of current and future efforts, the following sections will address the importance of the current trends in the evolution of traditional business models in relationship to SON with a detailed look at the entities that perform work, the work products, and the interactions.

The impact of Web based business models is being felt throughout the telecommunications industry as Web based companies push increasing volumes of traffic through service provider networks with service providers primarily being compensated based on access only. Bolstered by open source development, viral marketing, and fast development methodologies, Web based companies evolve quickly and reach directly to end users with innovative applications and content. In bypassing the traditional telecom model where the service provider "owns" the subscriber interface, the content, and the delivery methodology, the Web based businesses have essentially punched a hole through the service provider networks leveraging the networks for delivery while managing the end user relationship themselves. The resulting data traffic is driving increased costs for the service provider to deliver the bandwidth needed, but unlike SS7 based voice and SMS traffic, the service provider's revenues tend to be no longer tied to the volume of information traversing the network. The resulting gap between data costs and data revenue has led to a number of initiatives leveraging concepts from IT and telecom models to

develop new service provider business models including the current Web 2.0 and Telco 2.0 initiatives.

So the question becomes, how do the Service Providers approach the challenge and how does the SON framework support efforts to evolve the business models. One thing is now clear, users have evolved based on web usage and are savvier in managing multiple business relationships. We all have multiple user-ids and passwords with most users owning multiple devices with both fixed line and wireless access. Particularly with users between 18 and 30 years old, this has led to the decoupling of the device, portal usage, application usage, content delivery, and even partially the billing relationship from the service provider. Thus, while the service provider maintains control over access, user preference drives many of the other facets in the network activity. Potential business models are addressing many of these challenges along with the data usage while simultaneously evolving and supporting the voice and SMS business models. Some examples of business models which are emerging include the following:

- Tiered levels of service based on bandwidth coupled with support for specific applications supported by advertising revenue
- Tiered levels of service based on QoE guarantees with support for specific types of applications
- Network services provided to third party applications through APIs
- Content caching as a service
- Controlled portal with content outside the portal incurring additional charges
- Services integration (e.g. call waiting on your cable TV)
- Bundling – triple and quad play
- SDKs for third party development to expand application offerings

When viewing these business models within the context of the SON framework with consideration of the network architecture (e.g. NGN, IMS), the flexibility of the framework enables a clearer understanding of the interactions between entities and the interfaces needed.

Consider the following representations:

7.2.1 Example 1: Tiered levels of service based on bandwidth coupled with support for specific applications supported by advertising revenue.

Overview

Kyle is a gamer who plays online with several international players and has purchased a middle tier carrier service which support QoE guarantees for VoIP and Video but does not support QoE guarantees for gaming. Kyle initiates a VoIP call through the carrier's portal and the initiates a gaming service directly through his broadband access. The

carrier supports the VoIP call with monitored QoE, attempts to up-sell QoE for gaming and then manages the network when Kyle doesn't respond.

Tiered levels of service based on bandwidth coupled with support for specific applications supported by advertising revenue service flow

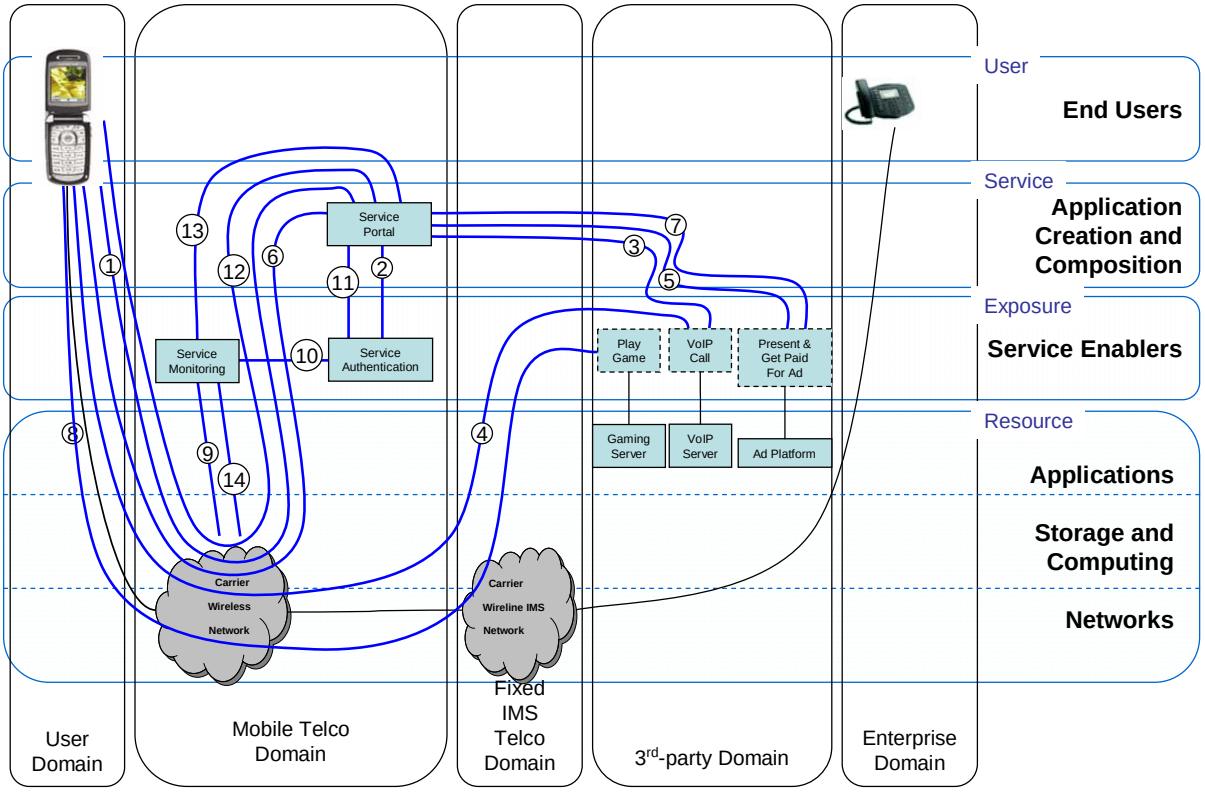


Figure 7-1: Tiered Service Example

Detailed Steps

1. Mobile user leverages a carrier hosted VoIP calling service to make a call to an international enterprise user.
2. The service portal triggers to a service authentication service enabler to verify the user's level of service.
3. The service portal initiates a VoIP call with a third party service.
4. The VoIP call service initiates a VoIP call between users.
5. The Service Portal invokes a 3rd Party Web Service to select the advertisement
6. The Service Portal renders the advertisement to the end user
7. The Service Portal requests payment for the advertisement being presented to the user
8. During the call, the mobile users initiates a gaming service directly with a third party provider
9. A service monitoring platform identifies the gaming activity

10. The service monitoring platform triggers to a service authentication service enabler
11. The service authentication enabler triggers to the service portal when the user's service level does not support guaranteed QoE for gaming applications
12. The service portal renders a message to the mobile device asking if the user wants to upgrade his service to provide guaranteed QoE for gaming
13. When the user doesn't respond, the service portal initiates a message to the monitoring portfolio that QoE is not guaranteed for the user's gaming activity
14. The monitoring portfolio communicates with the routing network to manage service levels

In the context of the SON framework, the scenario covers interactions between domains in which the carrier acts as a user recognized intermediately and in which the carrier interjects themselves into the activity in order to better manage the network and user experience. In each case, the framework provides a mechanism to represent the interactions supporting the business model.

7.2.2 Example 2: Services integration (e.g. call waiting on IPTV)

Overview

Kyle places a call to Kendall who is watching a program on her IPTV system. Kyle's calling information is displayed on the TV screen and Kendall uses her remote to accept Kyle's call.

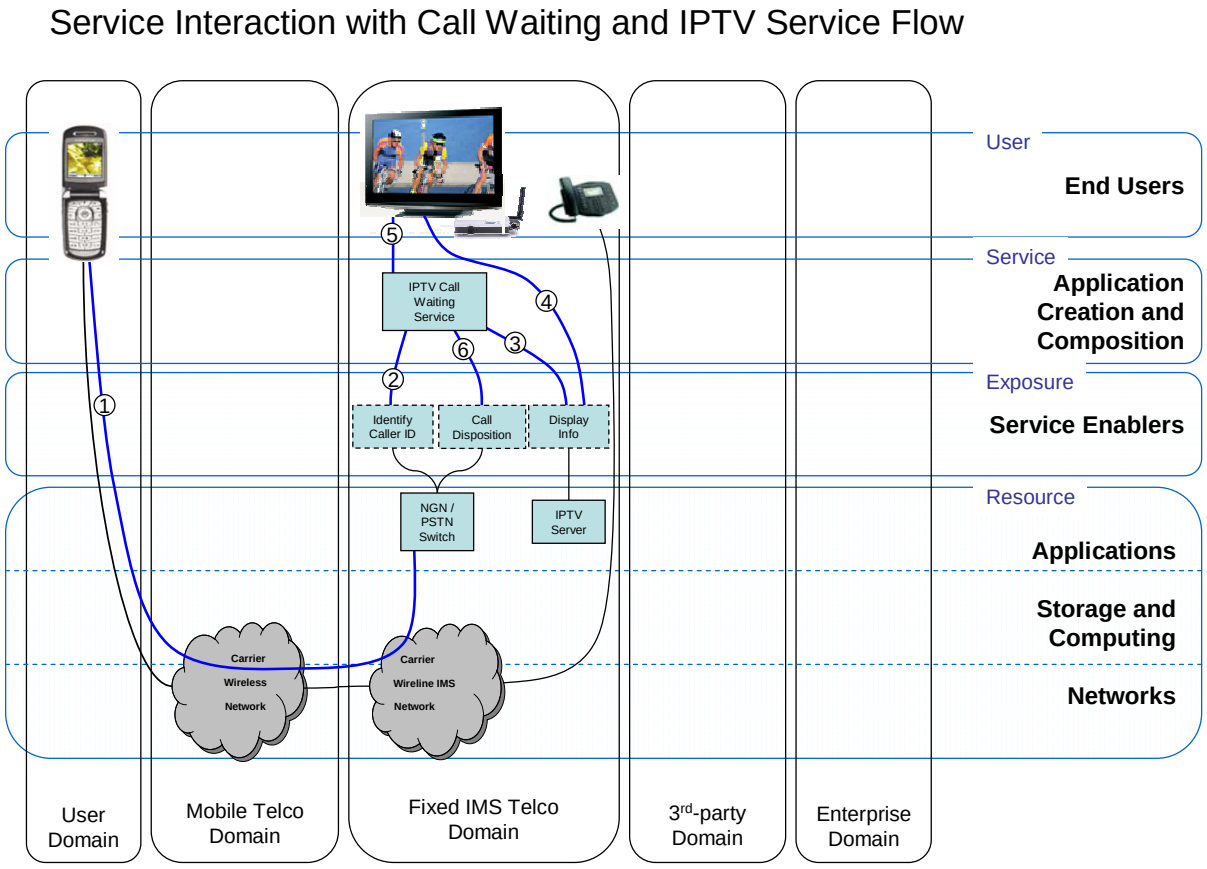


Figure 7-2: Service Interaction Example

Detailed Steps

1. Mobile user places a call to a wireline user.
2. The caller identification service enabler triggers an IPTV call waiting service.
3. The IPTV call waiting service looks up the caller's information and passes the information to the IPTV display service enabler.
4. The IPTV display service enabler leverages the IPTV server to display the caller's information and requests disposition for the call.
5. The user provides input as to the disposition of the call and this information is sent to the IPTV call waiting service application.
6. The IPTV call waiting service application forwards the disposition information to the call disposition service enabler.

   Depending on user input, the call disposition may be to connect the call in which case the switch would then connect the call.

Leveraging the SON framework, the scenario illustrates intra-domain interactions in which the carrier maintains the functionality providing service integration as a value added service within the business model.

The examples throughout this document illustrate the flexibility of the SON framework to represent interactions between domains and entities within those domains. While technically feasible to create interactions in unlimited combinations, the documentation of these interactions should be in support of targeted business models and should not in and of themselves dictate the business models being adopted.

7.3 The Impact of Virtualization on Business: Horizontalization

Virtualization of "things" leads to encapsulation of these "things". Such an encapsulation is beneficial to streamline organizations – in the terminology of Telecom players we are talking of transforming product and service "stovepipes" into a "horizontalized business layers" of an AssetCo, NetCo, ServCo, and SalesCo.

At Chief Information Officer (CIO) and Chief Technology Officer (CTO) level, other industries such as finance, utilities, logistics, or automotive are using the term "Virtualization Oriented Architecture" to classify new and changing business requirements of such a horizontalization and develop strategies and action plans how to gain or maintain their competitive advantage.

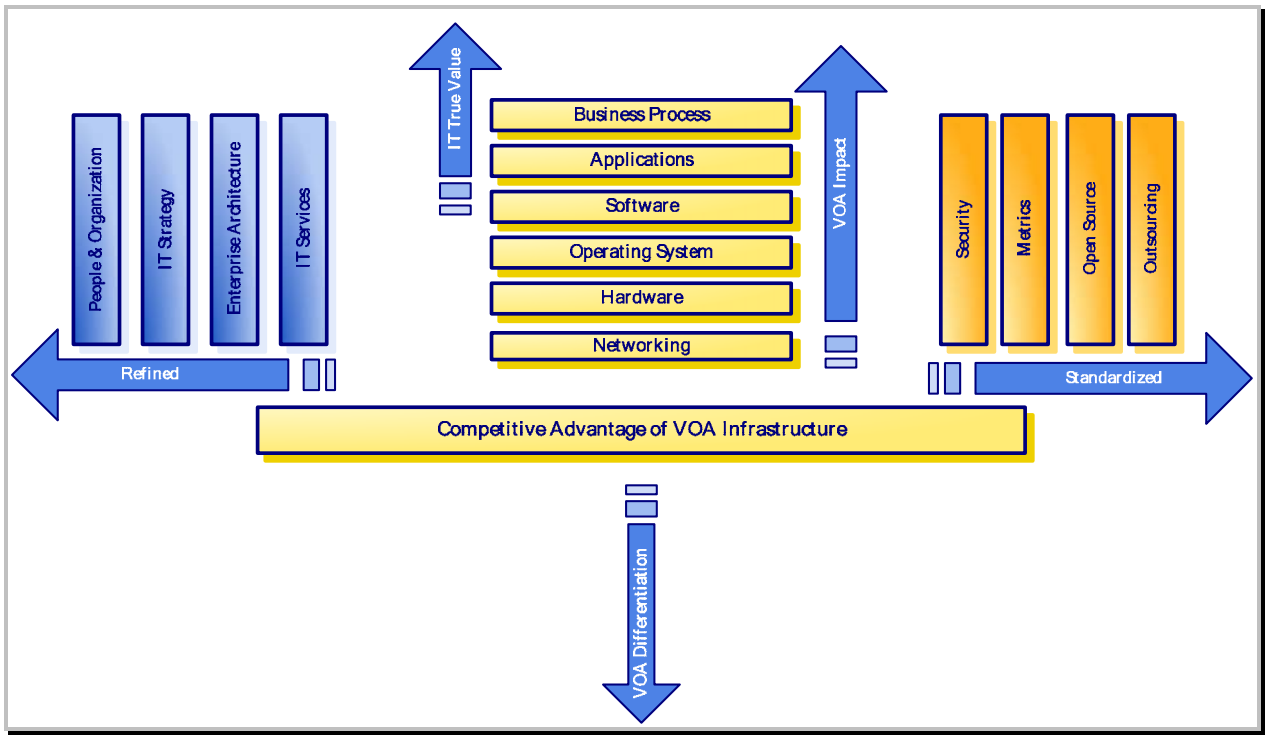


Figure 7-3: Transformational impact on business of a Virtualization Oriented Architecture (VOA) [Source: Wachovia]

1. **Supply and demand dynamics within companies will increase and become fickle.**
    - Virtualization will allow inter-exchangeable multi-sourcing, resulting in very personalized 1:1 as well as commoditized 1:n relationships – for consumers and businesses.
    - Lower capital and contractual entrance barriers will lead to faster uptake and

higher churn.

- Erratic usage fluctuation will challenge traditional ICT service provisioning and planning.
- Risk management, dynamic pricing, and flexible, automated SLA negotiations with other virtualized service providers will become essential.

2. **IT's true value will shift from Infrastructure provisioning to management and processes.**
    - Virtualization will lead to low-margin commodity resources for networking, hardware, and operating systems. Software, applications, and business processes for fickle demand markets will become IT's true value.
    - Common enabler functions of virtualization building blocks such as security, metrics, open source, and outsourcing will get standardized.
    - Management of people & organization, IT strategy, enterprise architecture, and IT Service planning will become key differentiators.
    - Console-And-Command-Line ICT administration will die. New key competencies will be capacity asset management, business-driven resource allocation, and automated real-time execution and operations.
3. **Modularization of IT-portfolios will foster process centricity instead of application-based deployment of ICT.**
    - Modularization of ICT offerings will increase customer menu options: Build on what you have, compose what you need.
    - Buying decisions will increasingly made by process owner.
    - Portals will mutate from static information to adaptable and interactive process platforms.
    - Formula based processes with intelligent documents (ID) will increase to foster open online and offline scenarios in heterogenic environments.

7.4 The Transformational Impact on Business

Markets and resources will be more in flux than before. Provisioning of product and service portfolios as well as the services and products themselves will change rapidly, though often only in small incremental steps. In order to gain or maintain a competitive advantage – or to stay in business at all – all companies regardless of size will have to excel in:

- **Forecasting of resource requirements and capacity:** processing power, bandwidth, connectivity, storage, data repositories, etc
- **Management and allocation of resources:** minimum reservations, limits, potential and required supplements, economics of resource mix, etc.
- **Execution and management of operations:** management of business services, application services, data services, Enterprise Service Bus, Server / Data / Messaging frameworks, and other infrastructure Services

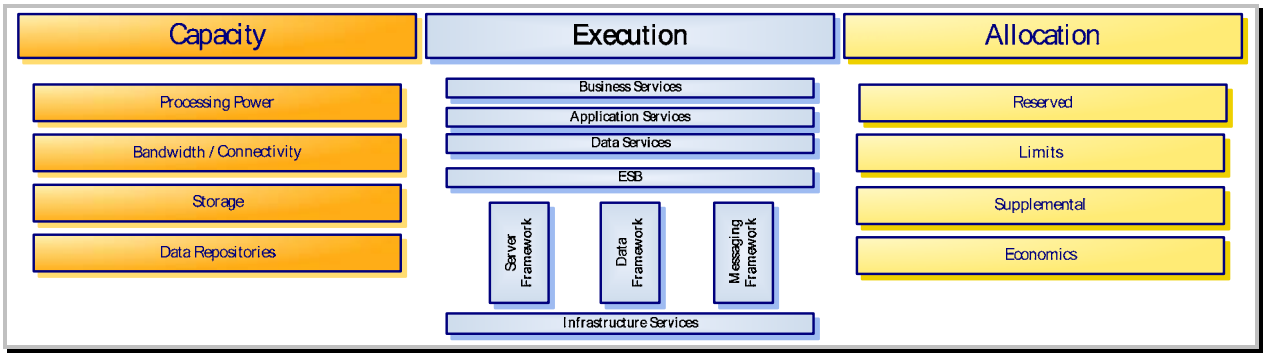


Figure 7-4: Transformational impact on business of a Virtualization Oriented Architecture (VOA) [Source: Wachovia]

4. **Accelerating product lifecycles will break the "watering can" ICT approach.**
    - The difference between application and service requirements of business units, divisions, and teams will increase.
    - Ever-changing Beta products will match the requirements of shorter plan-build-run-retire lifecycles and accelerated markets for some market segments better than tediously engineered applications.
    - The frequency of new application and service "deployments" will increase exponentially.
    - An increasing number of applications will be abandoned or retired before they leave the build phase.
5. **Actionable real time intelligence will become a key differentiator.**
    - Information logistics is and will be basis for most business decisions: Bringing the correct information at the accurate point of time in the correct format for the intended recipient at the right location.
    - Near-real-time delivery of high quality information will be driving enterprise applications.
    - Enabling real-time data sharing will be a key differentiator. Information-as-a-Service platforms and new service-oriented ways of accessing and integrating enterprise information are required to master these challenges.
    - Information lifecycle management will become a necessity to ensure the validity of information.

As a result, time allocation of CIOs will shift even further from focusing on IT operational excellence – the typical Function Head – to focusing on enterprise strategy, innovation, and differentiation – the typical Business Strategist. Operational excellence is still an imperative for successful business, but has to follow out of a grander business strategy, with execution left to business units and divisions.

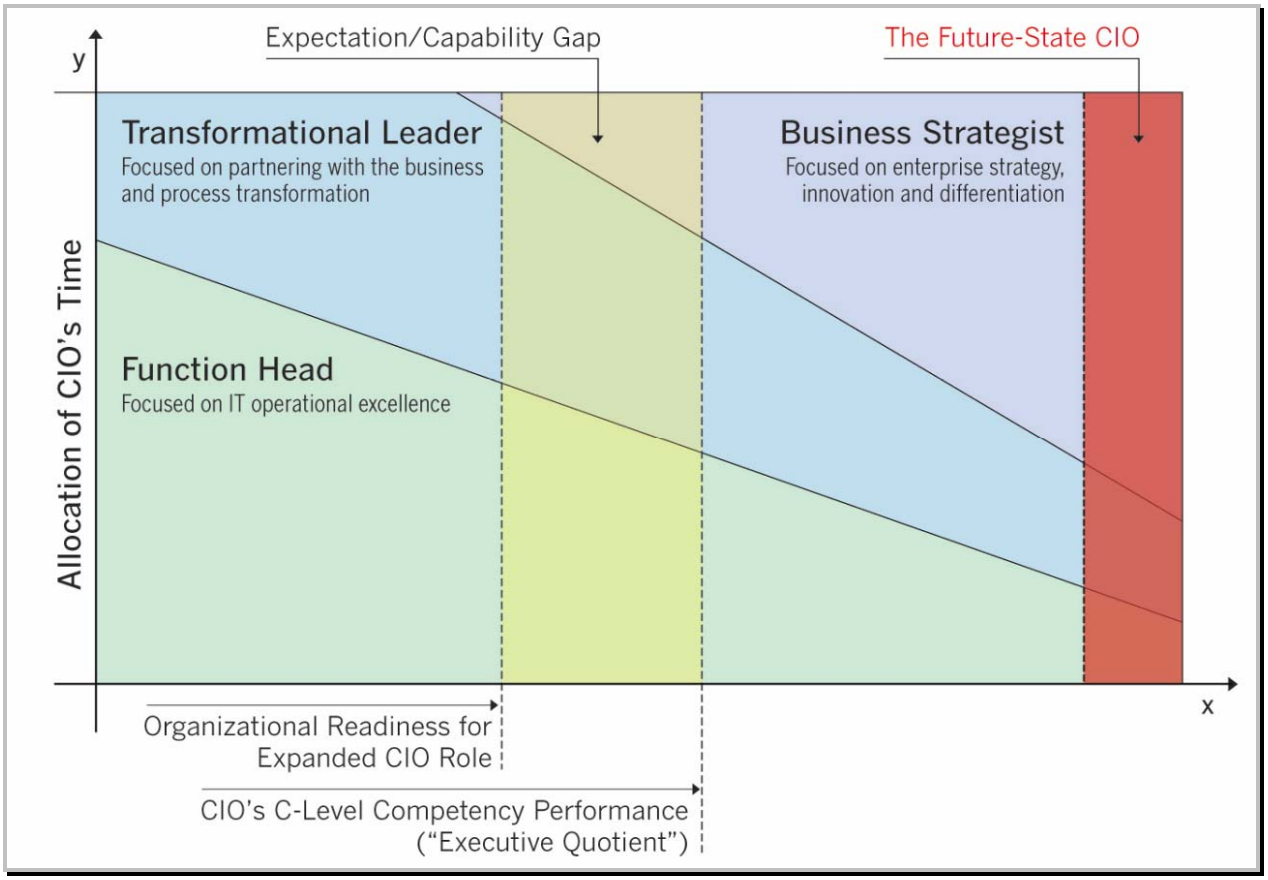


Figure 7-5: The Future-State CIO Model [Source: CIO Magazine]

This shift in mindset and responsibilities will have an impact on how applications and services are developed, provisioned, and operated; usage of services and applications will change; organizational structures and workforce will change.

6. **Employees will create web-based and mobile applications that may cause reassessment of IT department policies on applications use.**
    - Employees will utilize Web-based and mobile mashup concepts on existing business data to change user interface and data representation.
    - Mobile and nomadic devices will be used in personal and professional life, that may bypass or cause reassessment of control of applications and devices by the IT department.
    - The number of corporate and consumer ICT service portfolios and their versions will grow exponentially, greatly increasing lifecycle management complexity.
    - Employee-voted best-use cases will compete with corporate-dictated best-engineered cases.
7. **Personal and business life will mesh across company boundaries.**
    - Email and web-based social networking tools will traverse company boundaries. Personal social network contacts will be leveraged for business purposes and

vice versa.

- Mobile, nomadic, and teleworking will increase: the "office" will be wherever employees provide their workforce.
- Accelerating employee churn and personal experiences will constantly challenge and disrupt the corporate ICT service portfolio.
- Diffusion of ICT services, applications, and best practice will increase with employee churn rate and inside-out personal and professional contacts.

8. **Human Resources (HR) structures and required skill sets will become highly dynamic.**
    - Workforce will get decoupled from the number of assets, permitting significant staff reductions.
    - The break-up of static ICT solutions will promote a break-up of static HR structures.
    - Individual solution maintenance will rapidly decline. The demand for higher management skills of processes and policies will increase.
    - Just-in-time orchestration of virtualized infrastructure will become a lever with immediate and massive business impact, requiring a deep understanding of business mechanics.

7.5 The Transformational Impact on SON

On a high abstraction level horizontalization in Information and Communication Technology (ICT) industries will lead to a commodity infrastructure operated by an elite group of providers, while on the other end a plethora of applications will run on top of this infrastructure targeting more or less "niche" market. These markets might be summarized under umbrella industry descriptions such as "automotive", which actually includes thousands of markets and still only makes a minor percentage of the overall ICT market.

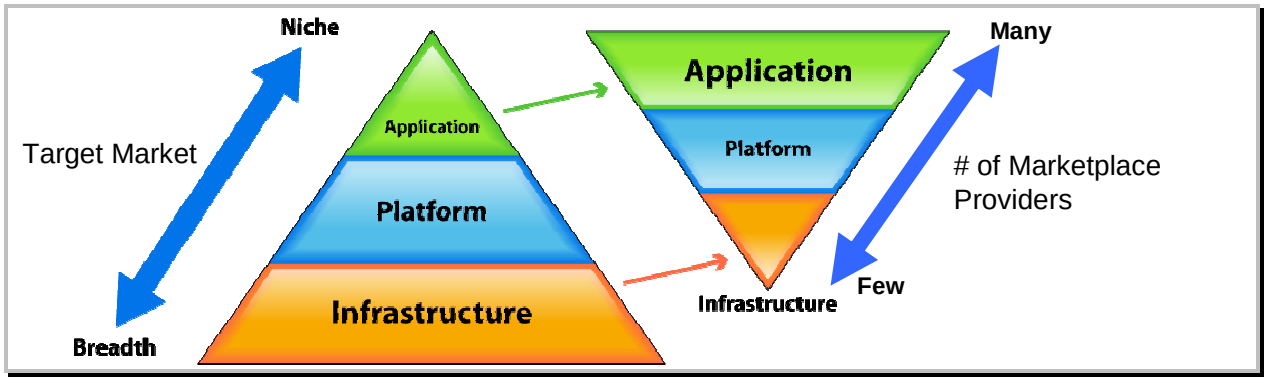


Figure 7-6: 1,000,000's of SON applications, dozens of platforms, an elite group of infrastructure providers [Source: GoGrid / ServePath]

In our SON Framework Model we already discussed how enablers are used to create new services and applications within different domains (user, provider, 3rd party). We also

mentioned the recursive character of the framework, as 3rd party applications can again used by other 3rd parties, and so on. When looking at SONs from a non-technical, high level business perspective, we will not think in terms of specific applications or the technical details of service enablers, etc.

Capacity, allocation, and execution management on a high level will mainly deal with terms such as suppliers, technology providers, partners, channels, solution providers, or integrators in order to ultimately provide services and applications either to the retail or wholesale side of a business. On this high level, CTOs and CIOs will think in the terms of "Extenders" and "Aggregators" that will interact with their ICT components.

Extenders provide additional basic functionality to applications, platforms, and infrastructure and usually heavily depend on very specific functionality or functional building blocks of the SON architecture. They often integrate legacy systems as well as emerging technologies, and usually become part of the SON pyramid over time.

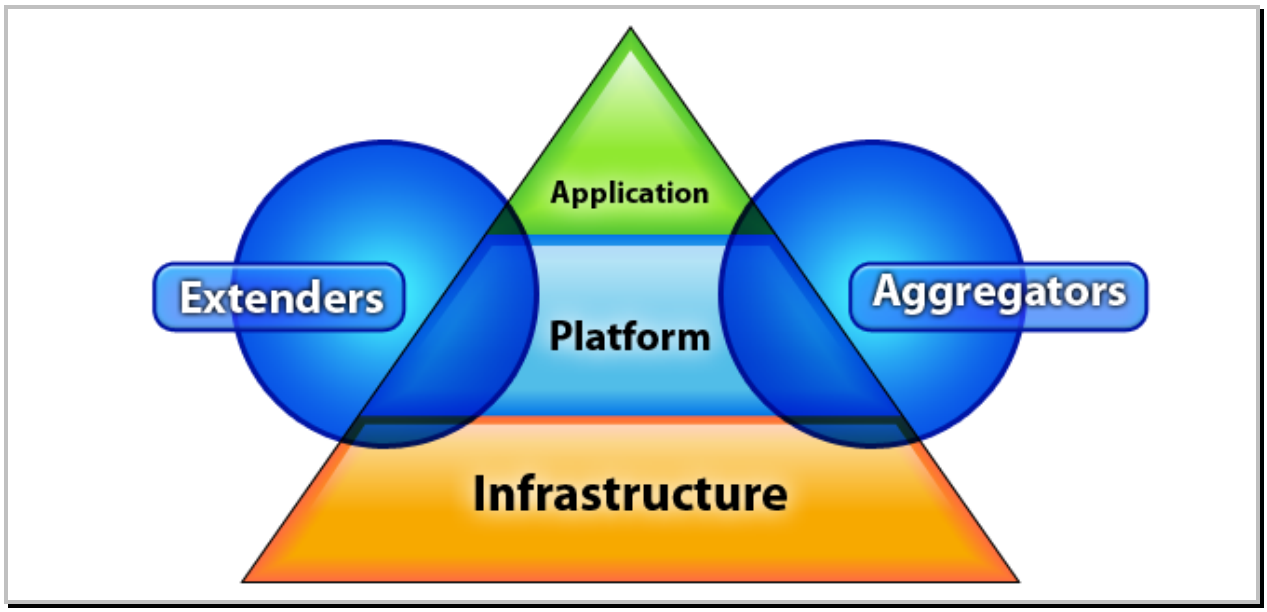


Figure 7-7: Special classes of extenders and aggregators [Source: GoGrid / ServePath]

Aggregators manage, integrate, or orchestrate several different applications, platforms, or infrastructure components for third parties. They heavily rely on functional building blocks of the SON architecture. Usually they will stay outside the SON pyramid and rather change suppliers, partners, technologies, or applications they aggregate.

Both extenders and aggregators can be internal as well as external to a company. In many cases the customers – who are not necessarily the end users – might again be CIO or CTO executives, with all the above said requirements and needs to cope with the transformation of their business.

7.6 3rd Party relationships, multi-sourcing

On the business relationship side, contracts need to be established that clarify the rules of engagement, operations, and disengagement. Part of that are terms of use, service levels, default options, as well as clarification of roles and responsibilities.

Difficulties can arise when an aggregator or extender is building its service on several enablers from different suppliers that all have different terms of use. This aggregator would then need to honor all (changing) terms of use, and might actually have to publish a superset of all of these terms to its users. Problems also might arise when terms of use from different enablers are conflicting or when the usage of the service the aggregator provides conflicts with terms of usage of its parts.

Another problem is trust in business relationships: Is it a problem if A trusts B but not C, and B suddenly sources an enabler from C without A knowing? A similar problem arises when legal or regulatory authorities prevent A sourcing from C, but B sources from C without A's knowledge.

> Note: This contractual level should not be a candidate for standardization within SON. While it is an important aspect, it lies in the hands of the service provider to comply with as well as enforce any contractual terms of use. If a company is aware of sensitivities within their business relationships such as government work or homeland security, it should be responsible for demanding disclosure of any harmful relationships of its third parties. Modeling of terms of use or policies itself can be done with TMF's SID. However, SID does not cover negotiation or event-driven evaluation of options or effects.

Business process design requires orchestration of application which in turn requires application descriptions, interoperability, required hardware to run on, software that an application needs to run, pre- and post-conditions, as well as operation assertions – what it does to a system of applications, when it terminates, and when it returns a sound result.

Furthermore, if an application by vendor A gets exchanged to an application by vendor B within these business process building blocks, it might not be desirable to immediately terminate all running instances of this business process, but rather let these processes run their course and only start new business process instances with that new software package.

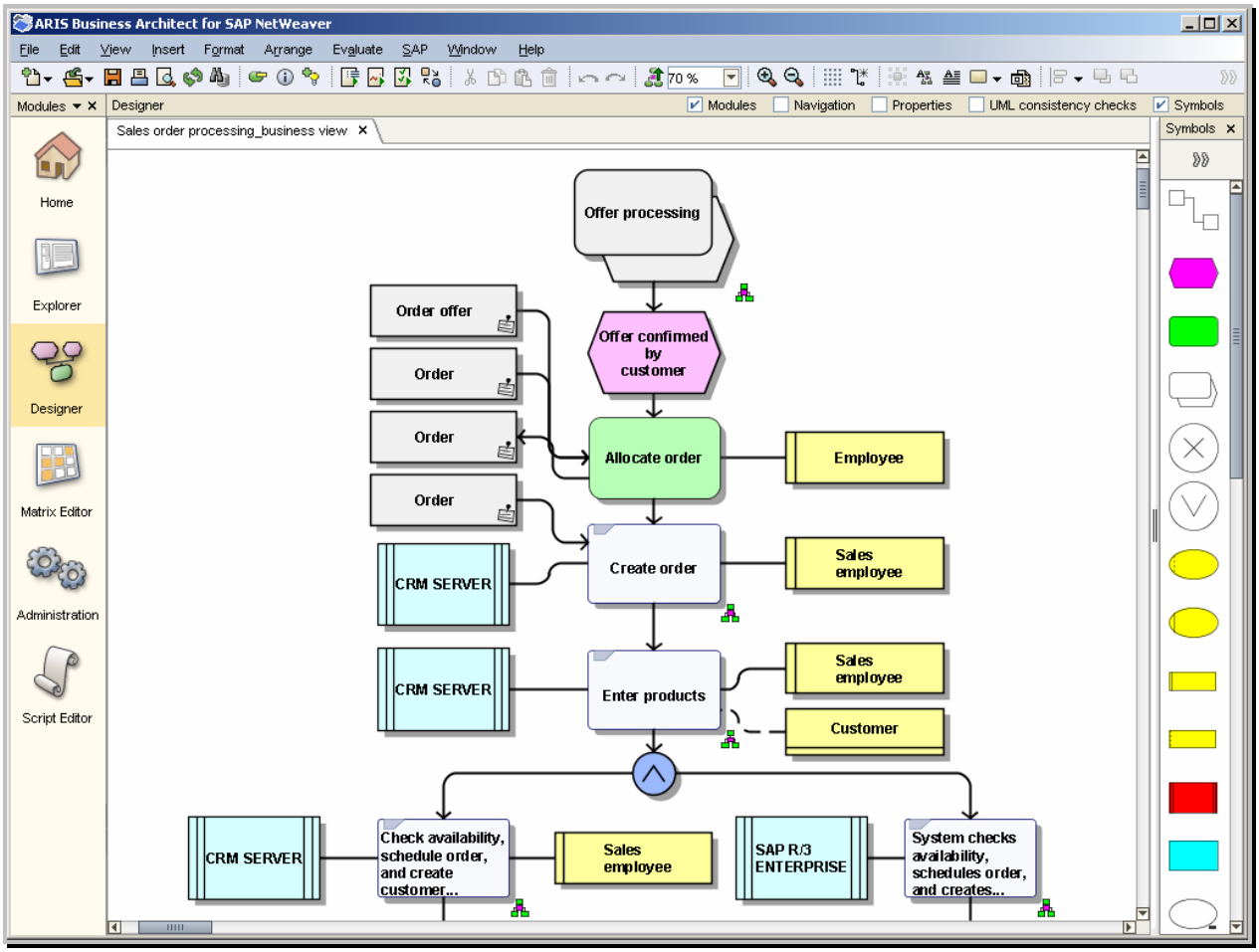


Figure 7-8: Business Process Orchestration with ARIS Business Architect for SAP NetWeaver at Polska Telefonia Cyfrowa ERA GSM (era.pl), a subsidiary of T-Mobile [Source: SchererIDL]

Note: Standard(s) should be used to describe functions, variables, and assertions to allow for (visual or semi-automatic) business process orchestration regarding capacity, allocation, and execution management. One important aspect that needs to be covered are expiration dates: dates until when a service enabler version is "guaranteed" to be available, services, maintained, and working.Potential candidates would be DMTF's Common Information Model (CIM) standards, TMF's Information Framework (SID) standards, or even OSI's Management Framework standards as a high-level framework. Neither one of them are intuitive, easy to apply, or have a steep learning curve, as Web x.0 kind of applications and services currently utilize and thrive on. That is simply because the subject of horizontal and vertical independence and exchangeability of applications and services is inherently complex. Web x.0 kind of applications don't cover that depth yet. They simply hope it works and constantly test APIs, applying user-driven patches as needed.

TDC's NGOSS self-service driven provisioning platform APT2 shows the complexity of business components that need to be able to interact, as well as the different technologies used.

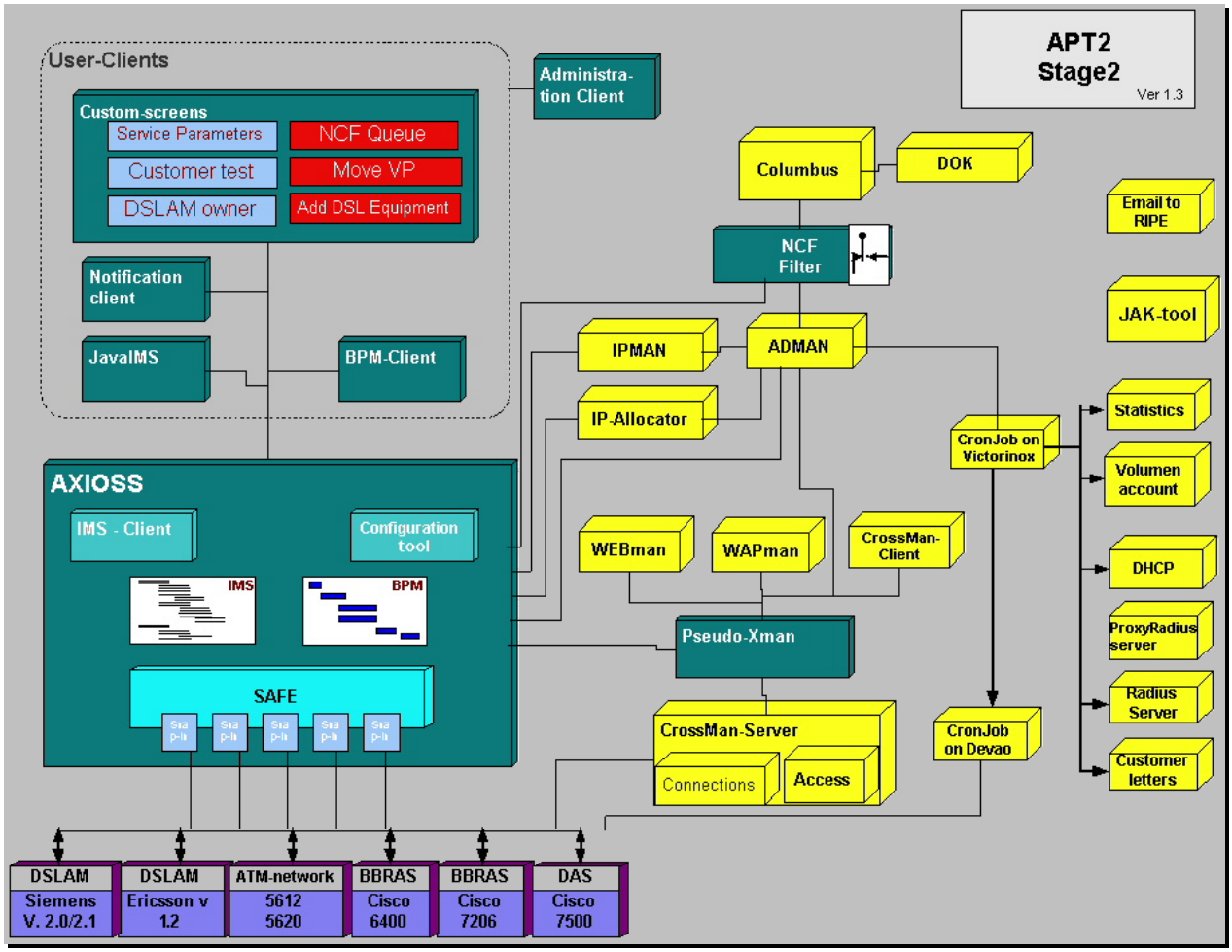


Figure 7-9: Complexity of New Generation Operation Support System (NGOSS) for automatic Asymmetric Digital Subscriber Line (ADSL) provisioning at TDC [Source: TDC]

Standards should also be available for communication of these data models between applications or business process building blocks. Current efforts at TMF as well as DMTF are not well developed yet. Business needs such as requesting a specific version, negotiating a version with a certain maximum computational power or delay, etc. are not yet defined.

One notion of "good" NGOSS design is the introduction of an intermediary between groups of business process building blocks: a "bus" system for integration. T-Systems' Customer Centric Fulfillment (CCF) architecture is such an example and is based on TMF's Application Map (TAM) and the Data Model SID. However, each building block is far from being universally exchangeable with other, similar software modules. The solution has a great degree of complexity with many customized integration parts to it. It is doubtful that any transition to an inter-exchangeable representation of all different software packets and service enablers would be trivial.

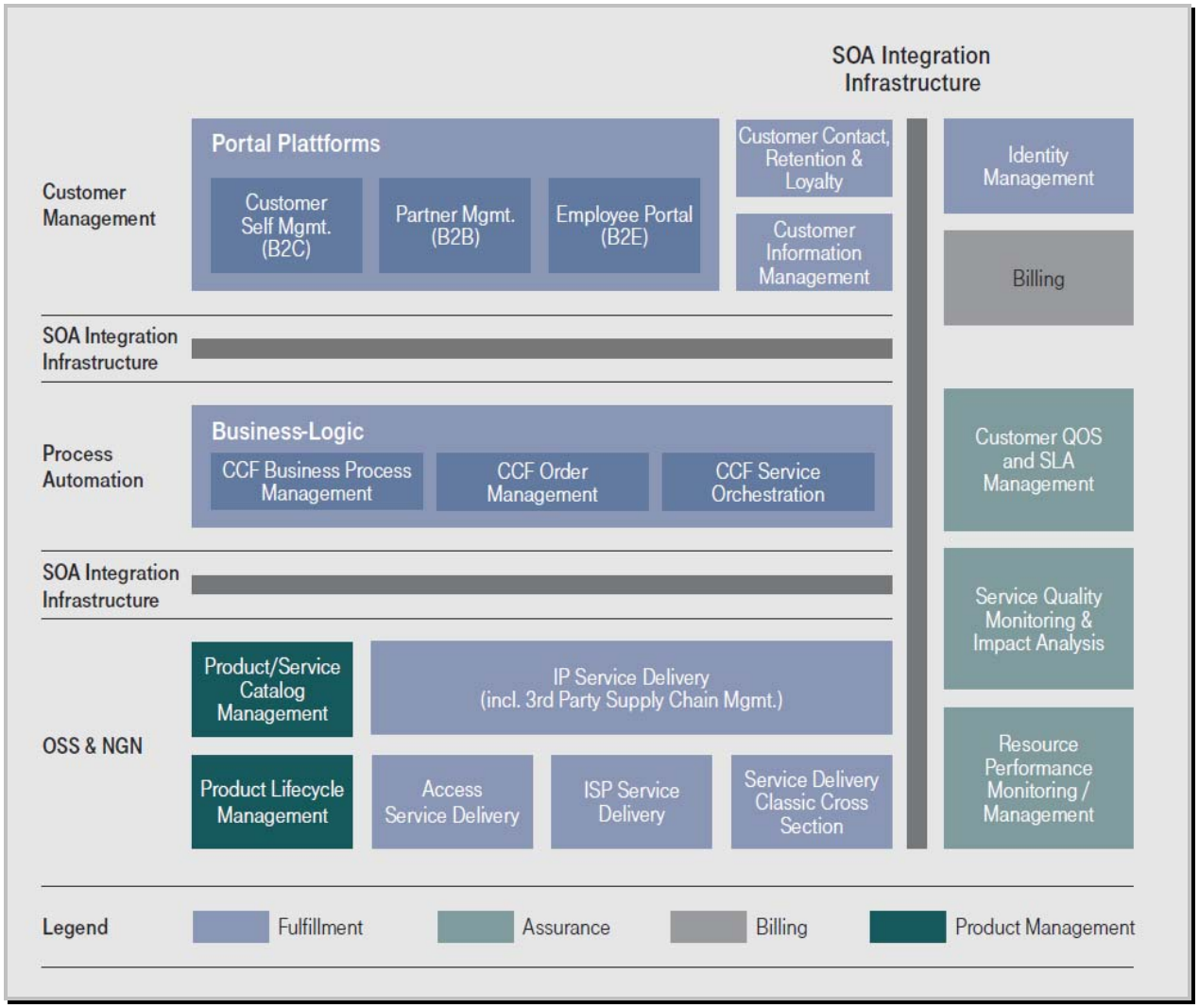


Figure 7-10: Customer Centric Fulfillment (CCF) architecture and solution at T-Systems, based on TMF's Application Map (TAM) and the Data Model SID [Source: T-Systems]

The approach for handling application instances that utilize several versions of different software packages and/or enablers is more an application design and IT architecture principle and can hardly be standardized.

7.7 Franchising, wholesale, partnerships, channel

"Franchisors" authorize the proven methods and trademarks of their businesses to "franchisees" for a fee and a percentage of gross monthly sales. Large media and communication companies as well as web shops see the return of (web) local stores. Franchises can help them for targeting advertising, localized service offerings, and local social networking and community building. Latest early trends in the US show increasing interest in franchising 4G wireless network and service providers – letting license owners operate independently, but under a great brand of the top three tier 1 mobile operators.

While traditional portals for advertisement and sales, better known as affiliates, are widely successful, with the arrival of new media, personalization, and location based services franchises become more difficult: Are the required services available in the franchise area? Is the quality the same? Can they be bundled in the same way? Are the data models comparable, and can they be used to compare sales data or predict usage trends in the same way?

Sales channels or integrator channels have a similar demand: groups of 3rd party service enabler users, aggregators, extenders, or clients are bundled into channels. Each channel has a distinctive characteristic, such as certain availability, help desk, service quality, feature sets. These characteristics need to be managed constantly.

For wholesaling mainly three aspects are of interests: A) Can a certain service enabler or service be decoupled from other service offerings as well as from existing billing relationships? B) What is the impact of wholesale traffic onto ongoing capacity, allocation, and execution management? C) Are legal and regulatory requirements for the wholesale offering met?

> Note: Supervision of features and characteristics are part of the SQM section of the document. It is unclear how well one could define a test suite or requirements catalog that could be run against a certain application or service in case underlying 3rd party interfaces or service enablers change… for a just-in-time replacement this might be tough to do. Usually telecom services are so complex that changes in wholesale products and subsequent testing for regulatory or legal compliance are subject to an analysis of a team of subject matter experts, lasting from weeks to months.
>
> It is questionable whether anything of this – beyond the possibility of querying current service and quality features – should be standardized.

7.8 Information logistics and governance – real-time and non-real-time

Information needs to be governed: what part to share when, and when to terminate relationships and request deletion of data. It is currently unclear how transparent services need to be in different parts of the world regarding user data usage: Users cannot opt-in or opt-out of service enablers they don't even know about. What happens if a service enabler is dynamically replaced by a different one that a user does not trust to handle his/her data confidentially?

As we explored earlier, employees will start bringing personal devices into business domains, and employee churn will increase. SON service enablers and services might need to control end devices, users, and roles in order to govern information correctly. Such governance gets complicated for devices that are not always online – how do you address a service's information security without hampering its usability?

> Note: These issues are not sufficiently addressed with any existing standard, neither Liberty Alliance standards nor other "open" standards. The problem always starts with

certification of "things". A certificate requires a common trusted root. To create such a root requires some efforts, and in effect there will be costs. Currently the complexity and cost point of certificate authorities does not work well together with SON business models of very short life times, lots of certification, and constant change.

7.9 Service development

The following sections address similar material. For each section, we will identify business entities that perform the work, what the work product is, and how the work is completed. The material is organized as follows:

- Service Development builds components.
- Service Orchestration integrates components at the Service and Exposure Layers.
- Service Provisioning binds network and customer data to infrastructure so that the system can be used.

7.9.1 Service Development (Business & Process)

Service development is about building stuff. It includes building logical stuff like services and service enablers. It includes building physical stuff like networks, storage, and processing.

The SON is different from the Public Switched Telephone Network (PSTN) in that a business entity can skip this step. The reality is that most businesses develop some component of their environment, but it is no longer a requirement. If a business or user produces a mashup from existing functionality, they have effectively skipped to Step 2: Service Orchestration. Keep in mind that while it is possible to skip service development, this work is the foundation of the SON, and without it, there are no higher level capabilities.

In describing the development ecosystem, let's examine the use case below. The use case shows an ecosystem that delivers a finished communications product. I will differentiate between products, features, services, and service enablers in this example. Products and features are business wrappering that are used to describe and deliver the product to the end customer. Services and Service Enablers have been described previously in the document.

The product contains 5 features: Privacy Manager, Call Forwarding, Caller ID, Call Logs, and Call Event Pop ups. The first three features are delivered by a single service enabler. The Call Log and Call Event features are developed as separate service enablers. Four types of service interaction management are used within the ecosystem. The role of service interaction is discussed further in the next section.

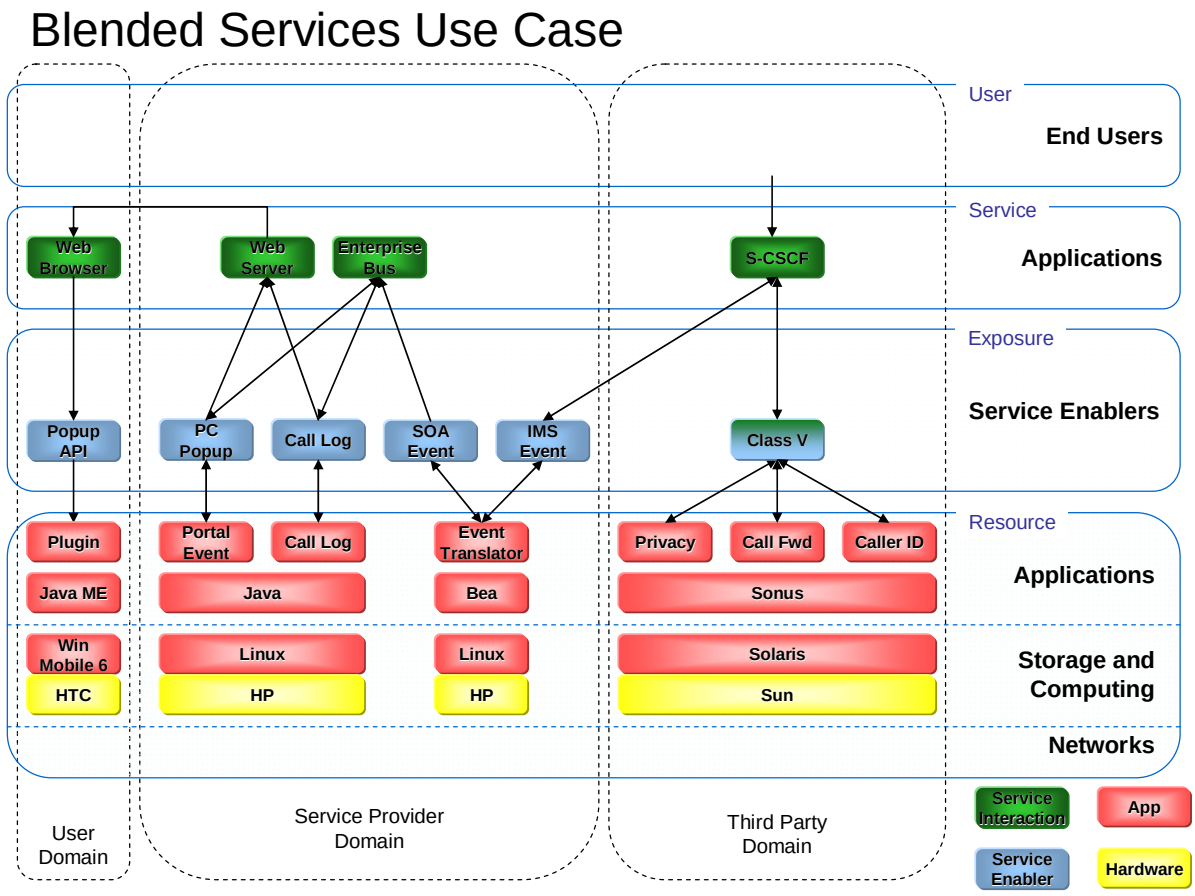


Figure 7-11: Blended Services Use Case

Within this use case we see multiple ways that service development can occur. Starting from left to right, each environment will be described.

7.9.2 User Domain

The user domain is a complicated environment where many components work together in the delivery of a finished product. In this example, HTML Component (HTC) is the prime integrator for all components except for the plug-in and pop-up API. The Service Provider contracts for the units with HTC and then adds the additional software as part of the fulfillment process.

7.9.2.1 **Storage and Computing**

In this layer, two suppliers work in concert to deliver the CPE. HTC is providing the hardware and embedded firmware. Microsoft is delivering the operating system through an Original Equipment Manufacturer (OEM) relationship to HTC.

7.9.2.2 Application

Again multiple suppliers deliver the functionality. Java Micro Edition (ME) is provided by Sun under an open source license agreement. The Opera browser is also provided an open source license agreement.

7.9.2.3 Service Enabler and Application Service

The service provider contracted for the development of the pop-up plug-in and API through a third party.

7.9.2.4 Service Provider

In the Service Provider environment there are two application servers running four service enablers. There is an additional server (not show) running the web server. The Enterprise Service Bus is a logical function that resides on the two application servers.

7.9.2.5 Storage and Computing

Both application servers are running a hybrid model on HP hardware and Linux software.

- In the Java environment, the service provider integrated Linux to some HP hardware from reuse. There is a software support agreement with RedHat. The hardware is supported by a closet of spare parts.
- For the BEA server, the HP server and Linux software was purchased as a package with a hardware and software support agreement from HP.

7.9.2.6 Application

- The first application server is running an open source Java environment. The service provider contracted for the portal event application in conjunction with the Plug-in and Pop-up API. The service provider's internal IT department developed the Call Log application.
- The second application server is running a commercial BEA environment based on Java. The Event Translator application was developed by BEA as part of a professional services agreement.

7.9.2.7 Service Enabler and Application Service

- The Web Server is an open source Apache implementation set up and supported by the Service Provider's eCommerce Team.
- The Enterprise Service Bus is an additional instance of a bus using the Service Provider's existing IT site license.
- The PC Pop-up and Call Log Service Enabler Packaging were developed in conjunction with the corresponding applications.
- The SOA Event and IMS Event Service Enablers were developed by BEA professional services as part of the Event Translator application.

7.9.2.8 **Third Party**

7.9.2.8.1 Storage and Computing

The Sun and Solaris environment is purchased through Sonus as part of a Class V application server bundle. Support is provided through Sonus as the integrator.

7.9.2.8.2 Application

The application is environment is delivered by Sonus. Sourcing of Sonus subsystems where applicable is shielded from the Third Party.

7.9.2.8.3 Service Enabler and Application Service

The Class V Service Enabler and finished service are provided by Sonus. The Third Party contracts to Sonus to have a small change made to the initial filter criteria for integration of the Service Provider's IMS Event Service.

7.9.3 *Sourcing Model*

The diagram below shows the previous functions by source. This shows the variety of sourcing models available in the SON.

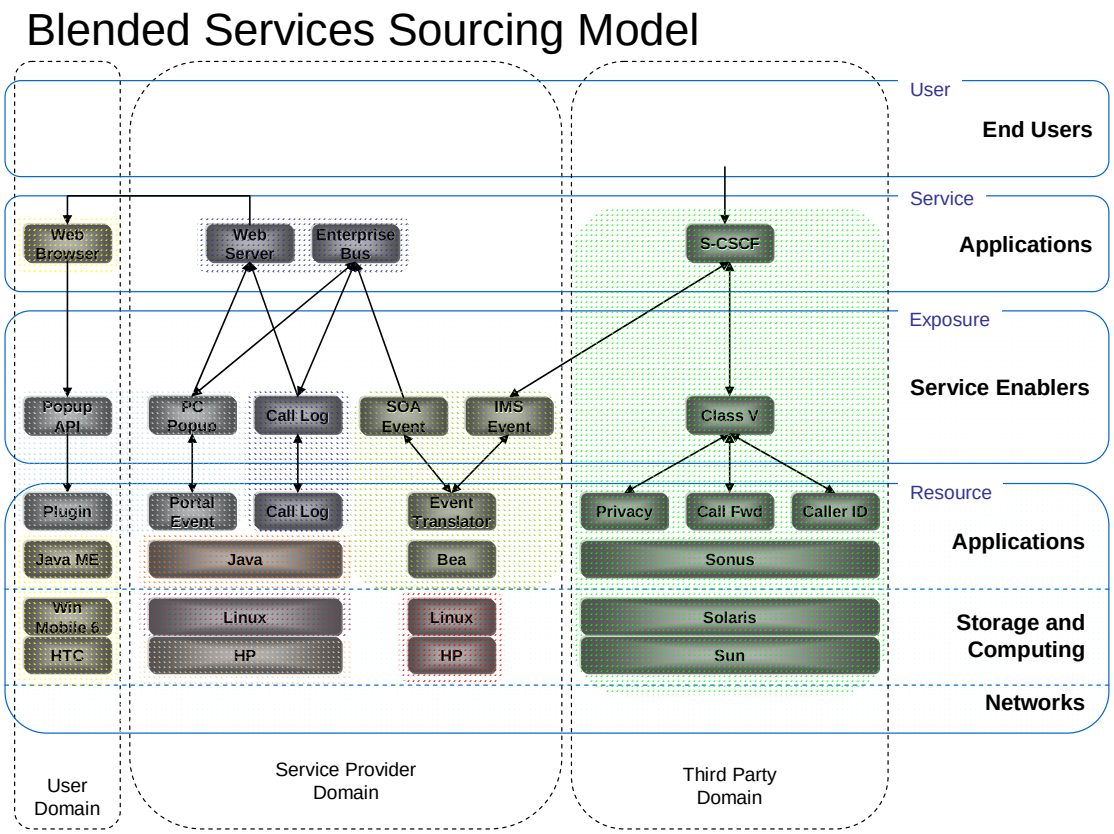


Figure 7-12 Blended Sourcing Model

7.9.4 Process Guidelines

Service Oriented Network development is flexible in both process and tools. From the framework, there are three layers: service (application), reuse (service enablers), and resource (application, storage, and computing). The framework also supports multiple roles including: user, service provider, and third party. Both the horizontal and vertical segments may be internally developed or externally supplied. Groupings may also be used to simplify development and integration.

In this flexible environment, it is important to establish a few guiding principles:

- The binding between the user and user profile for a specific service is singular and occurs within the service provider.
- The irreducible reusable component within the SON is the service enabler.
- Service enablers should be constructed using industry standard or open source application resources.

7.10 Service Orchestration (Business & Process)

Service Orchestration in the SON follows the model of globalization where multiple entities work together to deliver a single finished product. This is very different from the PSTN environment where service interaction was managed within a single supplier system within a single carrier. The SON model allows for business entities that deliver highly-specialized, modular services.

The sourcing model above shows how service orchestration is used across suppliers and across domains to integrate different components. The Call Session Control Function (CSCF) is used to bridge the Third Party and Service Provider Domains. The Enterprise Service Bus bridges components sourced from BEA, IT, and an external contractor. The Web Server also integrates with multiple externally sourced components.

The business entity assumes a specific vertical role in the SON framework. Within this role, the entity delivers modular capabilities corresponding to the horizontal layers of the SON Framework.

8 STANDARDS ACTIVITY ASSESSMENT

This section reviews the coverage of the standards and industry organizations that are considered relevant to Service Oriented Networks (SON).

8.1 Current Standards Development

Current Standards Development removed for external transmittal.

8.2 Standards Conclusions

SOA is the paradigm of building software applications based on orchestration of loosely coupled services into larger composite services. The advent of SOA tends to promote a new kind of software architecture where services, exposing features accessible through highly standardized protocols, are composed in a loosely coupled way. This paradigm shift is instigating a change of the role of the Telecom companies from pure telecom providers to service providers delivering communications-enablement of a wide variety of services, ranging from connectivity to existing management and IT applications.

Web Service technology is the prominent technology, but not the only technology, to deliver SOA. This work has been primarily accomplished in OASIS and W3C. Additional work remains to be done, particularly to adapt this stack and extend it to meet the needs of the telecom community.

8.2.1 Standards Landscape

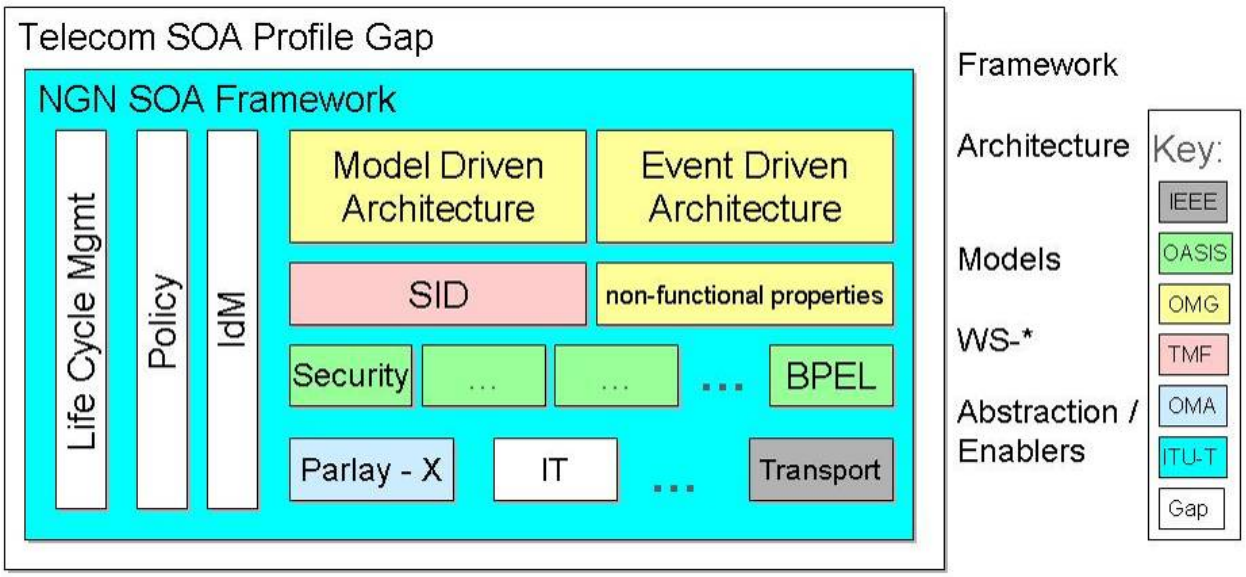


Figure 8-3: :SON Standards Landscape

In the current SOA environment, there are serious mismatches between the requirements of the IT world and those of the telecommunications world. Initially industry efforts were focused on meeting IT needs and it is only recently that attention has started to focus on meeting the needs of the telecommunications industry. Mainly these are incremental requirements (e.g. include real-time services, real-time service composition, asynchronous interactions, SOA-oriented integration of network management capabilities, enabling provisioning of service level agreements, distributed policy enforcement, testability, end to end security enforcement and verification and scalability). This additional functionality is needed by the telecommunications industry relate to the characteristics of telecommunications services needed by the telecommunications service provider so as to be able to reliably offer customers a guaranteed level of service in return for a specified payment. In the Telecom industry, these agreements are referred to as service level agreements (SLAs).

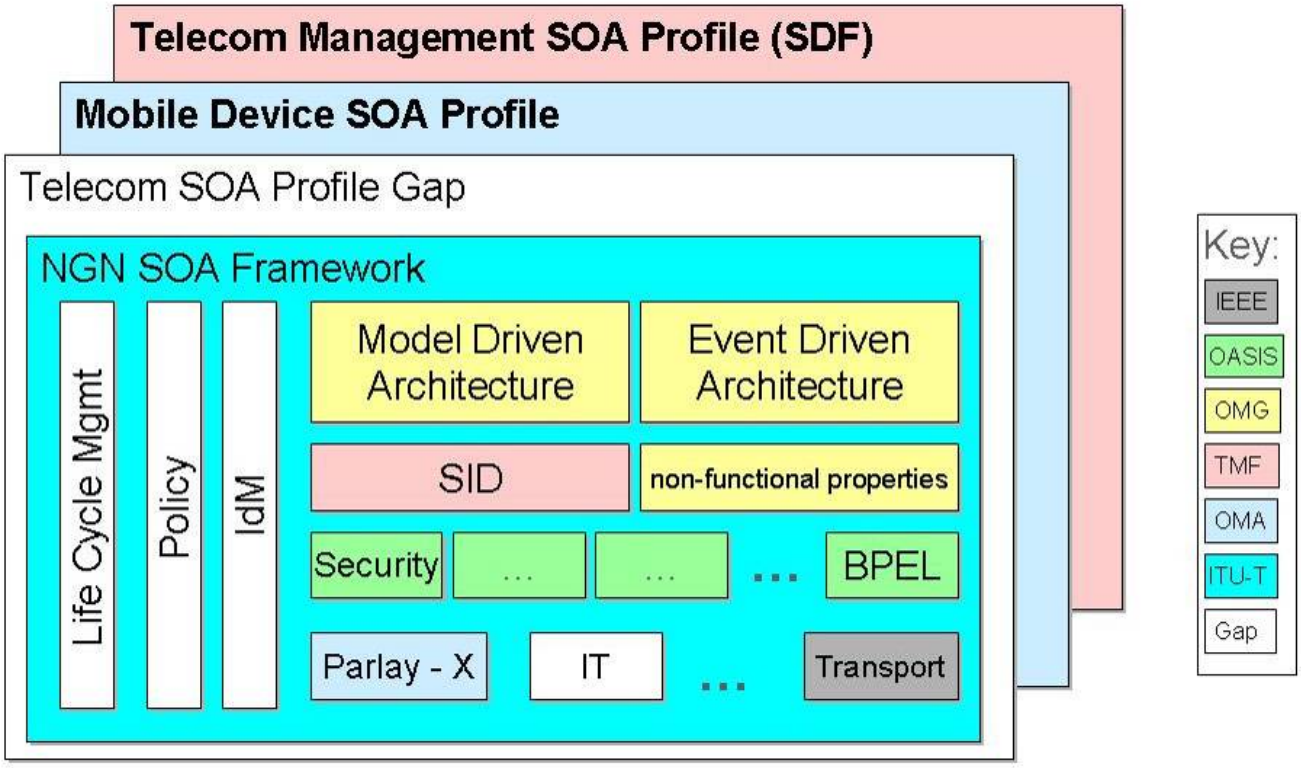


Figure 8-4: SON Standards Gaps for Telecoms

Existing standards currently provide a fairly stable, complete and converged IT-centric SOA/Web Services stack. There is now a requirement to build on this work by focusing on Telecom-centric standards extensions for the IT-centric SOA stack. The main objectives in this phase include the following:

- Develop required Telecom extensions of WS stack as the enabling technology for SOA.
- Develop standardized and complete Identity Management solution.
- Security.
- Develop required Telecom extensions of models and meta-models for services, business processes, Model Driven and Event Driven Architecture SOA enablement.
- Develop requirements related to the specific to expose NGN capabilities to be exposed to applications such as (e.g. Service Interfaces from network components and from support functions).

In particular to the areas in which ATIS may support standardization is the coexistence and migration of REST and Web Services.

The above sections provide an overview of a number of industry groups that are working on service-oriented networks-related activities. A number of these bodies and their activities warrant further investigation, monitoring, support and collaboration from ATIS in order to ensure the development of a cohesive standards environment for the SON. Key SDOs with whom immediate and close collaboration is needed include the ITU-T, OASIS, OMG, OMA and TMF.

9 GAP ANALYSIS

Gap Analysis removed for external transmittal.

10 CONCLUSIONS

10.1 General

There is a paradigm shift occurring where the stalwart institutions of communications services are being challenged and blended, requiring interworking, strategic development, and openness to a variety of concepts developed outside of the traditional Telco realm. As we have surveyed the other organizations and bodies of work affecting the SON, it has become clear that we are performing a unique exercise in thoroughly assessing the area from the perspective of the telecom industry. The issues are global in nature, and truly expand across traditional Telco, IT, and business domains. Piece parts of the puzzle have been developed in a variety of organizations, some largely unfamiliar to the Telco industry. We are in essence reaching a point of truth, where the hype of IMS and Web Services and SOA are subsiding to the realities of the limitations of each system, and the recognition that the strengths of each must be leveraged for a successful future.

The future of the development of the NGN, in which ATIS has played an important role thus far, will lie in the ability of the Telco's to utilize the vast resources of IT effectively, securely, and logically. This means the development of standards is needed, though in the near term, the reality is that proprietary options may exist, for example in the way services are delivered within a service provider's domain.

A cornerstone of the ATIS strategy is the notion that applications will have built-in communication capabilities and will become rich, highly valuable, and transformational for customers when they become highly collaborative and when they interact with and leverage the power of the underlying network. That interaction requires adaptation of the underlying communication and networking infrastructure to create communications-enabled applications. Communications-enabled applications will improve productivity, making it easier for applications to connect with people, at the right time, to do the right thing. The full power of this transformation will only be achieved with a fully open, interoperable, and extensible standards based approach.

SON-FG has concluded that it is not possible to deploy a service-oriented network entirely based on currently available standards. There are major gaps in the coverage of standards across the broad landscape of SON. Where standards are in place, they are often inconsistent, incomplete, or non-interoperable especially across the Telco, IT and web domains. Moreover, there is no single SDO responsible for standards across the piece.

There are critical gaps in the standardization of:

- Common service delivery platform
- Agile service creation
- SE descriptions
- Functional and non-functional aspects (and interfaces) for inter/cross domain exposure/ discovery, use, composability, and publishing of SEs
- Security and liability aspects of SE creation, use, and reuse
- SLAs/policies for implementation, including service quality management and quality of experience for the user
- Common Data Models and Name Space
- Service Oriented Infrastructure and IT virtualization
- Methods for a federated information architecture, common product catalogues, common metadata repository, and flexible billing and charging systems.
- Agile service wrap, including the construction of operational systems and processes from SEs
- Common policy reference information data model, and a common language (syntax and semantics).
- Service syndication
- Cross domain user profile data acquisition

One of the greatest challenges for communication service providers (CSPs) exists in the convergence of communication services with those available on the Web. The technologies, development practices, service philosophies, and business models are markedly different between these two worlds. We observe that traditional forms of communication (such as voice calls) are being displaced with message-based interactions around social communities (forums, blogs, social networks, etc.), especially for the young. As a result, some CSPs are seeing revenues fall and are witnessing a new generation entering the marketplace for whom the traditional Telco is not on their radar.

One way to meet this challenge is for CSPs to integrate Web-based services with those of communications, providing additional value along with reliability, security, and a common user experience. Currently, there are no standards in place to facilitate this and, in general, Web-based service providers (WSPs) display no desire to engage in the development of standards to this end. Another way for CSPs to meet this challenge is to present their services via open APIs for third parties to develop services. However, such APIs are not currently attractive to the main WSPs (e.g. Google, Yahoo, Facebook, etc.) who simply see CSPs as providers of the last mile of communications. Instead of convergence, we are seeing substitution.

The opportunity this presents for ATIS is to lead the telecom side of the equation in the effort to create dialogue between the IT space and the Telecom space, and to coordinate the development of standards particular to the ability of the Telco's to create, reuse, and deliver next-gen services. These include the ability to manage back end systems to support services and applications from across domains, the ability to develop and bring services to market efficiently, and the ability to reuse essential sources of information.

To support the above, recommendations to the ATIS organization to build upon the conclusions made in the ATIS Convergence report to specifically address the application and development space, the recommendation is made for ATIS to establish a forum to coordinate the development of standards for SON and to implement measures to provide the proper data extensions and schemas for implementing ATIS' standards in a SON environment. In addition a number of areas needing standardization and coordination with other SDOs and less formal groups have been noted and recommendations made for ATIS committees to liaise.

In order to make this convergence a reality, CSPs must assess, and where appropriate adopt, best practice from WSPs. This will not necessarily be restricted to traditional telecom standards methods. The proposed forum should also consider approaches to standardization, where complimentary and appropriate, that are based upon agile development methods in which de-facto standards are quickly developed and adopted by participants and/or the use of communal resources as in the opensource model.

10.2 Service Creation and Delivery

In order to deliver products and services in a SON environment, the requirement exists for an open environment (i.e. service delivery platform) in which the services are constructed from component parts. The development of the SDP is critical to the ability of Telco's to apply SOA to their service creation process. To agilely create services, both service enablers and their components need to be easily discovered and integrated. It is recommended that in the long term, the industry move towards a web services environment for service enabler interoperability, though in the near term, interoperability will likely be dictated through the development of proprietary SDKs. The development of web services standards to deal with critical aspects of service to do with addressing, routing, trust, security, and orchestration will enable robust, reliable, and scalable SONs.

The development of standards for service enabler descriptions and a means of commonly providing and locating data about service enablers and their components is necessary to support numerous facets of service creation and delivery including security, functionality, billing and charging.

In order to support the "mashup" and other business models whereby third parties use and reuse resources (data, SEs) from a service provider, the development of standard interfaces are needed. Service providers must be able to expose a set of enablers, services, or other resources to application developers, and other third party vendors. However, this exposure must be made with proper considerations for security and accounting (i.e. is it free or must charging take place).

The management of services for quality aspects becomes a complex scenario in a converged (inter provider and inter domain) environment. The implementation and negotiation of QoS and QoE will require the development of measurements and associated SLAs.

10.3 Service Provider Service Strategies, Inter-dependency, and Standards Gaps

A key theme of the SON FG work has been to allow for the mode of inter-dependency – i.e., for a Service Provider to utilize business entities and resources outside of the SP's domain for providing end-user services. That theme is contrary to the primary mode, historically, of telecom SP operation where the SP has relied only on itself and its own resources, soup-to-nuts, to provide end-user services, including service creation, testing, engineering and deployment, lifecycle management and operations support. That self-reliance has helped ensure high reliability, predictability and integrity of the end-user service offerings. If a SP's goal is to preserve the same level of service rigor, the mode of inter-dependency is essentially a frontier, presenting serious challenges in such areas as discovery, security, capacity, coherent management, revenue models, and trustworthiness.

The inter-dependency mode is certainly not new and is recognized as beneficial. It is employed extensively in the web world – (already to some extent a SON environment). It is seen in the evolution of cellular add-on features. It is emerging in video offerings that involve content acquisition, management and related advertising. However, the SON FG work places an even greater emphasis on it.

Key points are then, if relying on inter-dependency:

- Service Providers need to be very clear on the nature of services/features they want to offer to different classes of end-users and how they expect to generate revenue. SPs need consider the level of service rigor to be offered, how and what they want to brand or franchise, their degree of accountability to the customer/user if something doesn't work, and what the mechanisms are for generating and collecting revenue.
- SPs need to understand the technical standards gaps (which now should be a subset of gaps listed in this document) that prevent them from meeting their service strategy goals; and help drive standards accordingly. Meanwhile, inter-dependency can be a perilous, or at best a highly proprietary and much more costly, proposition.

Note also that the level or quality of service may be tied to what kinds of revenue strategies and therefore mechanisms a SP may choose to employ. For example, if a customer is directly paying for a service, say as opposed to support through advertising, the customer may have higher expectations for a high grade of service.

10.4 User Domain

Web 2.0 teaches the industry that attractive services require coping with user experiences (consumers or enterprises) which often today rely on user domain environments. However, innovative and lucrative services do depend on the ability to offer these applications from a well managed service delivery environment where quality of experience can be controlled. To reconcile these needs, SON applications should be both about opening up access to SP's network and enablers but also about access from SP's (hosted) applications to 3rd party and user domain environments and service enablers.

Applications that can be monetized will span multiple organization domains and require capabilities and behaviors that are not all located or dependent on network-side service enablers. Thus to keep a value-generating role for offering service applications, Service Providers must be able to leverage a consistent service delivery environment and set of programming models that can compose, orchestrate service enablers from the user (or 3rd party) domain environment as well. This requires open definition of service enablers from such domains.

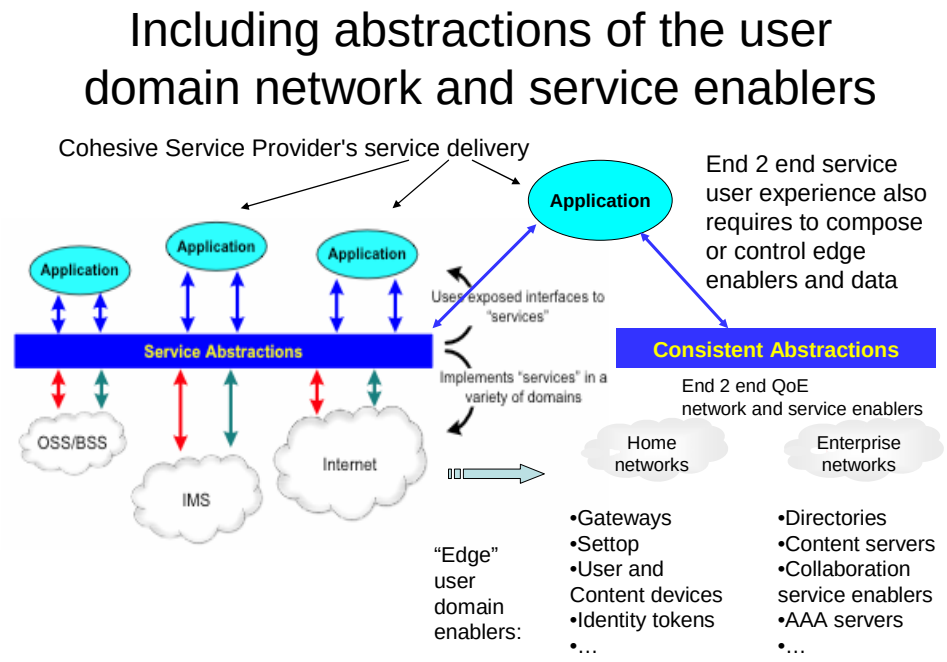


Figure 10-1: SON Applications

10.5 IT Infrastructure Virtualization

The development of SONs must be accompanied by critical developments in the IT space such as virtualization, storage, grid computing, and infrastructure to support a highly flexible environment in which resources can be leveraged effectively. The identification and support of

a standards developing body to provide leadership in the area of service oriented infrastructure is necessary for the long term success of SOA based services by Telcos.

10.6 Policy

Policy and policy implementation are an important aspect of SON and are key components of implementing a rich subscriber personalization environment. Service Oriented Network must include a clear description of how any service will support subscriber and network policies. The SON will also require the exchange of subscriber information across services and service enablers. Policy can be used as a service enabler to perform SOA composition and create new functions. The OMA PEEM should be considered as the basis of policy implementation within the SON, though it is not without need of enhancement to mitigate issues with its policy pattern implementation. In addition, more work is needed on how policy rule engines need to receive information from the network as input to the policy decision process. A presence based approach to enhance the function of the PIP is proposed and should be considered further for industry adoption.

APPENDIX A DEFINITIONS AND TERMS OF REFERENCE FOR SON

Note: Where applicable these definitions are harmonized with those found in the ATIS Exploratory Group on Convergence Report and Recommendations.

Service

The word 'service' has many meanings in the telecommunications environment, even as relates to the topic of SON. It should be clear in this document through the surrounding context and/or accompanying words what is meant in any particular instance of its use.

For our purposes, the two basic uses of *service* are:

a) Something of value being provided to and consumed by end-users. It may be a specific service, such as Verizon FIOS IPTV Premier or Acme Restaurant Finder, or it may be general, such as VoIP telephony service or emergency services.
b) A function that is performed by software whose output is used by some other entity. It may be, for example, a presence engine, or a routing function, or a number translation capability, or protocol interworking. 'Service' may also refer to the software itself.

The word 'service' is used in numerous industry terms such as:

- Service Provider
- Service Convergence
- Service Creation/Service Creation Environment
- Service Delivery Platform/Service Delivery Framework
- Service Oriented Architecture
- Service Oriented Network
- Service Broker
- Web or Web 2.0 services
- SaaS

Each may imply a more refined or qualified use of 'service' in its context. Some of those terms are further described below.

The ATIS Telecom Glossary treatment of 'service' firstly cites the Open Systems Interconnection Model definition, which is not terribly useful for SON FG purposes, and secondly cites the sense of (b) above [says specifically "This term is used as a generic reference to distributed applications that provide "services" to other applications", which is self-referential. But even within ATIS usage does not always conform to those definitions. Other SDOs or SDO-like bodies (OMA, TM Forum, 3GPP, etc.) may have their own definitions or usage patterns which may not be aligned. Likewise, internally, different companies may have their own perturbations on definitions or usages. It is not our purpose to reconcile those definitions/usages.

Examples of services in different contexts:

- It is comprised of multiple network or IT resources that are organized and behave according to the service definition in order to render a well-defined subscriber experience.
- A service can be understood as an application offered within a single domain and relying on composition at the Resources layers mostly.
- A service can be turned into a Service Enabler when complemented with appropriate logic and exposure building blocks, so that it can be reused or combined by another application through well-defined functional and operational interfaces.

Service Convergence

Service convergence is a sufficiently similar service experience from the user's perspective provided across different provider networks, access technologies, and end devices, recognizing limitations of different devices and technologies. Converged services can leverage shared application functions, across different access media, terminal types, and service providers. An example of service convergence would be the same service delivered to a mobile terminal, a television, or a computer with a sufficiently similar experience. These devices could be connected, for example, over a wireless data link, over a wireless voice link, over a co-axial cable plant, over a fiber plant, or over twisted pair.

Operational Convergence

Operational convergence provides maximum functional commonality or interworking of Business Support System (BSS), Operations Support System (OSS), network management, service provisioning, billing, et cetera across all service types (data, voice, video and any combination thereof) within the context of a Service Delivery Environment (SDE). The TeleManagement Forum (TMF), for example, has defined a NGOSS architecture that is applicable.

Quality of Experience (QoE)-as it relates to Service Quality Management

An important aspect of service is QoE. QoE requirements define the overall, subjective performance at the services level from the perspective of the end user. The establishment of consistent, baseline QoE for end users and the corresponding objective engineering targets is critical to the market success of many service offerings. When the user experience is within defined bounds, the service is deemed to be operating as expected within its QoE profile. A key direct measure of QoE is the Mean Opinion Score (MoS). Many objective measures, such as the duration of periods of degraded service (e.g., Degraded Seconds, Errored Seconds, Unavailable Seconds), may provide easily-measured parameters that can provide inferences to QoE, but may not measure all aspects of a given user experience of the service. QoE is also studied and formally defined in several places, including the ITU-T Study Group 12 (Q.13/12), in the ATIS IIF, QoS Metrics Task Force, and in the Broadband Forum Technical Report TR-126.

Since consistent QoE is recognized as a critical factor in providing a successful service, it is important to quantify the relationship between QoE and QoS. The reason that this relationship is important is twofold: it allows measurement of QoS to predict an expected QoE; and, given a target QoE, it allows the derivation of required service layer performance. This relationship is actively being explored in several standards forums, and the ATIS IIF (Issue IIF-2006-020) aims to create a concise and effective set of QoS measurements that would allow prediction of QoE for IPTV. The relationship between QoS and QoE may be expressed by means of one or more models and computations for video, audio, multimedia, and transactions.

In the context of SON, QoE shall encompass an end to end approach that spans the multiple Service Enablers used to render the ultimate user experience.

The QoE depends on how Service Enablers and their underlying resources deliver the right association of QoS, both from a network and IT standpoint. These resources may as well be located in adjacent domains, still QoE shall be considered end to end. For instance IPTV QoE may depend on resources in the core and home networks, as well as how the intermediate (and spread into multiple domains) software functions, like for instance media servers or settop boxes, behave.

Overall and in addition to network level QoS, an application (e.g. emergency alerting on TV, premium High Definition Television [HDTV] service…) shall also guarantee that the IT elements (application servers or intermediate enablers...) deliver on the right SLA or with the right set of (virtualized) computing and network resources allocated.

Applications must have the ability to give hints as to what the QoE requirements are in order for Service Enablers to set the right level of quality of service contracts with resources. Similarly applications must have the ability to adjust the user experience based on evolving characteristics exposed through the Service Enablers (e.g. an event triggered upon roaming between a hi-speed access network to low-speed one may require applications to adjust the user experience accordingly and automatically).

Service Composition

Service composition imparts to the user the experience of one seamless, composite service instead of separate services, by way of *additional functionality* that binds service enablers together. For example, IPTV and VoIP service where the viewer's IPTV screen displays "Mom" over the current program and the sound mutes while the incoming call from Mom is delivered to the target device (e.g., house phone). Or as another example, a caller's yellow pages web query for nearby Thai restaurants leads to a reservation confirmed with the chosen restaurant with caller preferences (e.g., no smoking), and the reservation and directions downloaded to the caller's Personal Digital Assistant (PDA)/calendar.

Service Provider

Service provider convergence allows continuity of service across multiple service providers. This is intended to include services incorporating mobility and nomadicity, which includes roaming between providers. For example, service providers may provide access, network connectivity, applications or content.

Security

Security covers all aspects of security, including network access, media access, media distribution, software upgrades, billing exchange, and application authorization. In addition, security must consider how all these individual security mechanisms can be combined to achieve end-to-end security.

To be successful, effective security policies must be developed and implemented in a systematic, consistent, and rigorous manner for services and networks. Developing effective security policies is best achieved by using a comprehensive security model such as the ITU X.805 Security Architecture. The X.805 Security Architecture allows for a structured approach to developing security policies and determining what security services need to be deployed.[33]

User Profile

User Profile is user specific information related to communications, multimedia services (e.g., Home Subscriber Server (HSS), User Profile Server Function (UPSF)) and applications (e.g., Universal Description Discovery and Integration (UDDI)). User Profile also includes user policies applicable to multiple access media, different service providers, and different presence personalities. The User Profile can be accessed securely with its subsets able to be exchanged securely as required, including for billing and management purposes.

Device Profile

A device profile is device specific information which identifies the capabilities of a device. This can include information on embedded applications and firmware as well as device configuration and ideally contains information on upgrades, updates and version numbers to applications or physical device accessory attachments. The device profile is used in the context of service convergence to define the parameters which affect QoE, transfer/concurrence and security posture.

[33] ATIS Next Generation Network (NGN) Framework, Part I: NGN Definitions, Requirements and Architecture, Issue 1.0, November 2004, Section 2.5.2.

Quality of Service & Service Level Agreements (QoS/SLA)

QoS/SLA is the application of QoS management mechanisms and any applicable corresponding Service Level Agreements, to a given service, irrespective of the access media, terminal type or service provider. QoS encompasses the cumulative effect of numerous factors affecting service and network performance such as application availability and network transport characteristics. Typical QoS management in the IP domain involves the delivery of services within defined parameters for delay/latency, dropped packets, mis-ordered packets, jitter and error. QoS has end-to-end considerations across single or multiple providers.

Policy-based Resource Control and Management

Policy-based Resource and Control Management refers to a set of logical functions and physical components that enable the optimal operation and management of network resources. Policy Management provides policy control, resource allocation and management, and admission control functions based on pre-defined policy rules that can be associated with the access transport, session control, applications, peering, core transport and routing.

Service Subscription

Admission and control may occur at multiple layers of the system: network, call and session control and application layers. Network registrations enable a device to attach and originate, or terminate IP packet flows. Call control registration enables the ability to send and receive session requests. In the case of some applications, such as video and gaming, a subscription may be needed to gain admission or access to content services.

Each of these layers involves a type of registration service and may require authentication and authorization. Such an authorization may be accompanied by encryption keys to enable packet and message exchanges associated with a registration, session, or subscription to be continuously protected by authentication, integrity and confidentiality services. It is important to note that the time scales for each layer may be different. Network access may change every few minutes. Session registrations could last for hours or days. Subscriptions could last for months.

Having long-lived keys for subscriptions may enable the underlying multicast and broadcast services to be relieved of the need at session setup time to establish security between the terminal and the network and thus reducing setup time and enhancing the speed at which service continuity may be achieved during terminal movement. For example, a video terminal may need to simply download the broadcast guide, tune it and deliver content with little per-terminal state maintained by the network.

Service Provisioning and Monitoring

Service provisioning and monitoring is the provisioning to establish and revoke service, and the monitoring of service availability in order to maintain service levels.

Maintaining service quality requires the providers to communicate network characteristics and cooperate in delivering the service, be it for prevention, provisioning, growth, customer issue management or other purposes. The diagnostics can isolate the area of the fault in an end-to-end service, and possibly across multiple service providers.

Information Models

An information model is an abstract but formal representation of entities including their properties, relationships and the operations that can be performed on them. The entities can be either real or intangible.

Application

Like 'service', the word 'application' has numerous meanings across the telecomm industry.

For our purposes, the two basic uses of *application* are

a) Software that embodies the primary logic characterizing and supporting an end-user service and its features. Such an application may reside on an application server (AS) within a service provider's network or may be a 3rd Party application outside of a service provider network.
b) Software on user devices providing something of value consumed by the end user. E.g., MicroSoft Word or Firefox Web Browser or Google Maps.

Applications interact with other applications or enablers or other service-type software, and that interaction typically would occur over a network (or networks).

The ATIS Telecom Glossary treatment of application is "Software that performs a specific task or function, such as word-processing, creation of spreadsheets, generation of graphics, facilitating electronic mail, *etc.*"

Application Server

An application server is a physical computer that supports a runtime environment for applications. The system software is an intrinsic part of the application server. Application servers ease application development by providing a suite of system software building blocks provided by the application server.

Service Enabler

A service enabler is a function (or set of closely related functions) in the domain of a service provider that is exposed though a defined interface, toward other resources. Service Enablers are re-used through their interfaces. Service Enablers are separated from the applications using them and from the resources that they make use ofService Enablers become the basic building blocks for creating services.

A service enabler is a self-contained software function that can run autonomously and can also be composed with other enablers, as a component of a larger Application, to render an end to end user experience.

- An enabler is defined / exposed through several interfaces towards:
- applications for use and composition with other enablers
- end user or devices environments for use and composition (mashup)
- service delivery environments for lifecycle management
- resources, or other enablers, it makes use of, or cooperates with, in order to render the intended function, including network functions
- Optionally, operations and charging environments (note: open discussion item)

A service enabler can be application domain agnostic and commonly used by all applications, e.g.:

- user profile
- policy
- identity

A service enabler can expose a more domain-specific function reused by some applications, or in specific network environments, e.g.:

- messaging
- conferencing
- content delivery
- address book

A service enabler can be the result of a composition of other service enablers with additional logic that provides increased and reusable value to applications than if they did the composition on their own.

Application Enabler

An Application Enabler is usually associated with general purpose IT capabilities fully independent of the underlying service network. For application software running on an application server, an application enabler is the software building blocks that can be used to implement that application. It can be exposed and reused across Application execution environments.

Service Orchestration

Service Orchestration is a mechanism for doing dynamic service composition. In a service oriented architecture, a service orchestration tool (e.g. Web Services orchestration) can be used to create a combined service or integrated service (offering a combined or integrated user experience), by orchestrating the invocation of individual services, in response to user requests, network events or other inputs. A service interaction manager is one example of a service orchestration tool, although this term is normally only used when the orchestration is achieved by manipulating or processing network protocol traffic.

Interfaces

An interface represents a means of exposing the function(s) of an enabler for use by any resource. A defined interface tells the resource what services the enabler that offer the interface is prepared to provide. Interfaces only tell a resource how the functions of the service enabler can be used. The interface makes no assumptions on the resources that may use it.

Resource

A resource can be an application, component, function or enabler that can send, receive or process a request. An enabler itself is also a resource and can use the function of another enabler through its defined interface.

IP Multimedia Subsystem (IMS)

Definition

IMS is an architecture defined by 3GPP originally for GSM wireless networks for supporting multimedia services to end-users. The architecture is being expanded ~~by~~ through the involvement of other SDOs (e.g., ETSI TISPAN, CableLabs, 3GPP2, WiMax Forum) to include needs for wireline and other wireless networks and related user equipment, as well as service continuity across access network types. 3GPP, including ATIS as an organizational partner, is in the process of combining needs into a single "Common IMS." IMS is differentiated from previous architectures in that it natively supports IP mobility and association of services directly to users.

Examples

The following systems are examples of IMS Systems.

AS: Application Server
BGCF: Breakout Gateway Control Function
HSS: Home Subscriber Server
P/I/S-CSCF: Call Session Control Function
RACS: Resource Admission Control Subsystem (ETSI TISPAN)
SCIM: Service Capability Interaction Manager

IMS defines service capabilities, interfaces, and a session interaction model.

Next Generation Network (NGN)

Definition

A NGN is a packet-based network able to provide Telecommunication Services to users and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent of the underlying transport-related technologies. It enables unfettered access for users to networks and to competing service providers and services of their choice. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users. [ITU-T Recommendation Y.2001 (12/2004) - General overview of NGN]

Examples

The following systems are examples of NGN Systems.

- RACF: Resource Admission Control Function (Y.2012)
- NACF: Network Attachment Control Function (Y.2012)

Service Oriented Architecture

Definition

SOA is a software architecture where functionality is grouped around business processes and packaged as interoperable services that are used in concert to deliver an end user function.

SOA separates applications into distinct units, or services, which are made accessible over a network in order that they can be combined and reused. SOA concepts are often seen as built upon, and evolving from older concepts of distributed computing and modular object oriented programming applied on carrier scale environments.

Each "service" within the architecture provides a defined set of capabilities through a specified interface. These interfaces are usually based on web services. The aim is a loose coupling of services with operating systems, programming languages and other technologies which underlie applications.

SOA also describes IT infrastructure which allows different services to exchange data with one another as they participate in applications. These services communicate with each other over an Enterprise Service Bus (ESB) by passing data from one service to another, or by coordinating an activity between two or more services. The Enterprise Service Bus is also used within the SOA for Service Interaction Management. The ESB provides reliable message delivery and translation to relieve individual services of maintaining network topology.

SOAs are not standardized across ecosystems, but service capabilities are intended to be reused within an implementation domain.

Examples

The following services are commonly deployed within a SOA.

- User Profile
- Security Mechanisms
- Enterprise Service Bus
- Web Servers

Legacy IT services and network capabilities are frequently wrappered with web services for integration into a SOA. This wrappering can transform the legacy service or network capabilities into a service enabler.

Web 2.0

Definition

Web 2.0 is a term which describes the trend in the use of World Wide Web technology and web design that aims to enhance creativity, information sharing, and, most notably, collaboration among users. These concepts have led to the development and evolution of web-based communities and hosted web services, such as social-networking sites, wikis, blogs, and folksonomies.

In Web 1.0, the internet was a source of information (html & ftp) where consumers searched and consumed content on the internet. Web 2.0 added interactive functionality to the network and ability to assemble distributed applications. To that extent Web 2.0 is a distributed SOA platform implemented on the internet.

Tim O'Reilly first outlined Web 2.0 in his blog.[34] He specified the following guiding principles:

Strategic Positioning

- The Web as Platform
- User Positioning
- You control your own data

Core Competencies

- Services, not packaged software
- Architecture of Participation
- Cost-effective scalability
- Remixable data source and data transformations
- Software above the level of a single device
- Harnessing collective intelligence

[34] O'Reilly T. What is Web 2.0 [Internet]. Available from: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Examples

The following sites or categories are all examples of Web 2.0.

- Grand Central Station
- Google Maps
- Podcasts
- RSS: Really Simple Syndication
- SyncML
- Social Networking such as MySpace
- User Generated Content such as YouTube

Web 2.0 services can be used by themselves or combined with others to form higher level web applications (mashups). In *The World is Flat*, Thomas L. Friedman describes globalization. Specifically he describes how multiple companies act in concert to bring a product to market. Web 2.0 is globalization for the internet.

PSTN Service

Definition

Public Switched Telephone Network Services exist within a set of network elements including: Class V Local Switches, Class IV Tandem Switches, Signal Transfer Points, Service Control Points, and Intelligent Peripherals. These services can be broken into two categories including in-switch and AIN. In-switch services were contained within a single supplier solution. AIN services were developed in the mid-80s and included the definition of a call model that allowed multiple network nodes to interact in the delivery of a service.

Examples

The following sites or categories are all examples of PSTN Services.

- LNP: Local Number Portability
- 8XX: Toll Carrier Call Routing
- Call Forwarding
- Authorization Codes

Legacy PSTN services were developed prior to availability of SOA, and typically use proprietary service creation environments. Many of these services also use an SS7-based call model defined by AIN. Reuse of PSTN services in SOA will require wrapping the legacy service to fit the new framework.

APPENDIX B ACRONYMS

Acronym	Definition
.NET Framework	Software technology that is available with several Microsoft Windows operating systems intended to be used by most new applications created for the Windows platform
3GPP	3rd Generation Partnership Project
3GPP2	Third Generation Partnership Project 2
3GSM	Third Generation GSM services
AAA	Authentication, Authorization and Accounting
AAW	Application Aware Networks
ADSL	Asymmetric Digital Subscriber Line
AGW	Access GateWay
AJAX	Asynchronous Java and XML
AP	Access Point
API	Application Programming Interface
ARIB	Association of Radio Industries and Businesses
AS	Application Server / Autonomous System
ASAP	Agile Service Assembly Process
ASC	Agile Service Creation
ASOAT	Adopting SOA for Telecom-OASIS Technical Committee
ATIS	Alliance for Telecommunications Industry Solutions
BDT	Telecommunication Development Bureau
BEA	BEA Systems, Inc, acquired by Oracle
BPEL	Business process execution language
BPM	Business Process Management
BPMN	Business Process Modeling Notation
BR	Radiocommunication Bureau
BSS	Business Support System
CAG	Corporate Advisory Group
CAPEX	Capital Expenditure

CBCS	Categorization Based Content Screening
CC/PP	Composite Capabilities/Preferences Profile
CCF	Charging Collector Function
CCSA	China Communications Standards Association
CDMA	Code Division Multiple Access
CGI	Common Gateway Interface
CIM	Common Information Model
CIO	Chief Information Officer
CIOC	Chief Information Officers Council
ComSoc	IEEE Communications Society
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-The Shelf
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CRM	Customer Relationship Management
CROWN	Centralized Resources Over Wide Area Network
CSCF	Call Session Control Function
CSP	Communication Service Providers
CT	Core Network & Terminals [GSM Working Group]
CTO	Chief Technology Officer
CWM	Common Warehouse Metamodel
DCOM	distributed component object model
DDF	Disk Data Format
DMTF	Distributed Management Task Force, Inc formerly Desktop Management Task Force
DSL	Digital Subscriber Line
DSLF	Digital Subscriber Line Forum
DVD	Digital Versatile Disk
E2E	End-to-End
ebXML	Global framework for the use of XML in e-business data exchange

EDA	Event-Driven Architecture
EDGE	Enhanced Data rates for GSM Evolution
EERP	End-to-End Resource Planning
EGC	Exploratory Group on Convergence
EMMA	Extensible Multimodal Annotation Language
ENUM	Electronic Number Mapping
ERA	Polska Telefonia Cyfrowa
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
eTOM™	Enhanced Telecom Operations Map
ETSI	European Telecommunications Standards Institute
EU	European Union
FCC	Federal Communications Commission
FDD	Frequency Division Duplex
FiOS	Fiber Optic Service (Verizon)
FOKUS	Institute for Open Communication Systems (Germany)
FTP	File Transfer Protocol
GERAN	GSM EDGE Radio Access Network (GSM working group)
GGF	Global Grid Forum
GII	Global Information Infrastructure
gLite	ITU standard ITU G.992.2 for ADSL using discrete multitone modulation
GPM	Global Permission Management
GPRS	General Packet Radio Service
GRIA	Grid Resources for Industrial Applications
GridRPC	Grid Remote Procedure Control
GSI	Green Storage Initiative
GSM	Global System for Mobile Communication
GSMA	GSM Association
GSSM	General Service Subscription Management
HBA	Host Bus Adapter

HDTV	High Definition Television
HP	Hewlett-Packard
HR	Human Resources
HSDPA	High-Speed Downlink Packet Access
HSS	Home Subscriber Server
HTC	HTML Component
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
ICAP	Internet Content Adaptation Protocol
ICT	Information and Communication Technology
ID	Intelligent Documents
IDE	Integrated Development Environment
ID-FF	Identity Federation Framework (Liberty Alliance)
IDL	Interface Definition Language
IdM	Identity Management
IdM-GSI	ITU-T IdM Global Standards Initiative
IdP	Identity Provider
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IETF	Internet Engineering Task Force
IIF	IPTV Interoperability Forum (ATIS)
ILM	Information Lifecycle Management OR Infrastructure Lifecycle Management
IM	Instant Messaging
IMA	iSCSI Management API
IMI	Identity Metasystem Interoperability-OASIS Technical Committee
IMS	Internet Protocol Multimedia Subsystem
INCITS	International Committee for Information Technology Standards
IP	Internet Protocol
IPDR	Internet Protocol Detail Record
IPR	Intellectual Property Rights

IPTV	Internet Protocol Television
IRP	Integration Reference Point
iSCSI	Internet Small Computer Systems Interface OR SCSI protocol over TCP/IP (IETF draft standard)
ISO	International Organization for Standardization
ISpheres	Digital repository system that is designed to stand on its own or act as a front-end to existing databases
IT	Information Technology
ITU	International Telecommunication Union
ITU-D	International Telecommunication Union - Telecommunication Development Sector
ITU-R	International Telecommunication Union - Radiocommunication Sector
ITU-T	ITU-Telecommunications Standardization Sector
ITU-T	International Telecommunication Union- Telecommunication Standardization Sector
J2EE	Java 2 Platform Enterprise Edition
J2ME	Java 2 Platform Micro Edition
J2SE	Java 2 Platform Standard Edition
JCA-IdM	ITU-T Group: Joint Coordination Activity for IdM
JSON	JavaScript Object Notation
LBS	Location Based Service
LIF	Location Interoperability Forum-now part of OMA
LNP	Local Number Portability
ME	Micro Edition
MGIF	Wireless Village, Mobile Gaming Interoperability Forum—now part of OMA
MIDlet	Mobile Information Device toolkit, Java program for embedded devices
MMA	Multi-path Management API
MMSC	Multimedia Messaging Service Center
MMS-IOP	MMS Interoperability Group-now part of OMA
MOF	Meta Object Facility
MPEG	Motion Picture Experts Group
MSDN	Multi-service Data Networks

MTD	Metadata Committee (ATIS IPTV Interoperability Forum)
MTOSI	Multi-Technology Operations System Interface
MWIF	Mobile Wireless Internet Forum, now part of OMA
NACF	Network Access Control Function
NGN	Next Generation Network
NGOSS	Next Generation Operation Support System
NGSON	Next Generation Service Overlay Network
NWG	WiMax Networking Group
OASIS	Organization for the Advancement of Structured Information Standards
OASIS Telecom	OASIS Telecommunications Services Member Section
OBF	Ordering and Billing Forum (ATIS)
OEM	Original Equipment Manufacturer
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture
OMA	Open Mobile Alliance
OMG	Object Management Group
Open CSA	Open Composite Services Architecture Member Section of OASIS
OPES	Open Pluggable Edge Services
OPEX	Operational Expenditure
ORT	Operational Readiness Testing
OS	Operating System
OSA	Open Service Access
OSE	OMA Service Environment
OSF DCE	Open Software Foundation Distributed Computing Environment
OSS	Operation Support System
OVMF	Open Virtual Machine Format
OWL	Web Ontology Language
OWSER	OMA Web Services Enabler
P3P	Platform for Privacy Preferences
PC	Personal Computer

PDA	Personal Digital Assistant
PDP	Policy Decision Point
PDSN	Packet Data Servicing Node
PEEM	Policy Evaluation Enforcement and Management Enabler
PICS	Platform for Internet Content Selection
PIP	Picture in Picture
PKI	Public Key Infrastructure
PRQC	Network Performance Reliability and Quality of service Committee (ATIS)
PSTN	Public Switched Telephone Network
PTSC	Packet Technologies & Systems Committee (ATIS)
PTSN	Public Telephone Switched Network
PVR	Personal Video Recorder
QCI	QoS Class Identifier
QoE	Quality of Experience
QoS	Quality of Service
RACF	Resource and Administration Control Functions
RAID	Redundant Array of Independent Disks
RAN	Radio Access Networks (3GPP)
RAND	Reasonable and Non Discriminatory, refers to IPR
RDF	Resource Description Framework
REST	Representational State Transfer
RESTful	Services using the REST approach
RF	Radio Frequency
RFI	Request for Information
RMI	Remote Method Invocation
ROI	Return on Investment
RSS	Really Simple Syndication
RTES	Real Time and Embedded Systems
RTP	Real Time Transport Protocol
SA	Services & System Aspects (3GPP)

SaaS	Software As A Service
SAC	Service Access Code/Special Area Code
SAN	Storage Area Network
SCA	Service Component Architecture
SCE	Service Creation Environment
SCF	Service Control Function
SCIM	Service Capability Interaction Manager
SCS	Service Capability Server
SCXML	State Chart XML
SDE	Service Delivery Environment
SDF	Service Delivery Framework
SDK	Software Development Kit
SDO	Service Data Objects
SDOs	Standards Development Organizations
SDP	Service Delivery Platform
SE	Service Enablers
SG	Study Groups
SG17	ITU-T Study Group 17 on Security
SID	Shared Information/Data Model
SIMPLE	Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SMIL	Synchronize Multimedia Integration Language
SMI-S	Storage Management Specification (SMI-S)
SML	Standard Markup Language
SMS	Short Message Service
SMSC	Short Message Service Center
SNIA	Storage Networking Industry Association
SOA	Service Oriented Architecture

SOAP	Simple Object Access Protocol
SOI	Service Oriented Infrastructure
SON	Service Oriented Networks
SP	Service Provider
SPARQL	Simple Protocol and RDF Query Language
SQM	Service Quality Management
SS#	Social Security Number
SS7	Signaling System 7
SSO	Simplified Sign-On
SUP	Subscriber Update Protocol
SVG	Scalable Vector Graphics
SW	Software
SyncML Initiative	SyncML Initiative, Ltd., now consolidated into OMA
TAM	Telecom Application Map
TC	Technical Committees
TC-Grid	Technical Committee addressing the convergence between Grid and networks in ETSI
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDM	Time Division Multiplex
TISPAN	Telecoms & Internet converged Services & Protocols for Advanced Networks
TMF	TeleManagement Forum
TMF TAM	TM Forum Applications Framework
TMOC	Telecom Management and Operations Committee (ATIS)
tModel	A data structure representing a *service type* (a generic representation of a registered service) in the Universal Description, Discovery, and Integration registry
TMS	Telecom Member Section of OASIS
TNA	Technology Neutral Architecture
TOPS	Technology and Operations Council (ATIS)
TTA	Telecommunications Technology Association

TTC	Telecommunication Technology Committee
TV	Television
UAT	User Acceptance Testing
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
UE	User Equipment
UIM	User Identity Module
UL	Uplink
UML	Unified Modeling Language
UNICORE	Uniform Interface to Computing Resources
UPSF	User Profile Server Function
UPU	Union Postale Universelle
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USD	United States Dollar
UTRA	Universal Terrestrial Radio Access
VCC	Voice Call Continuity
VLR	Visitors Location Register
VOA	Virtualization Oriented Architecture
VoiceXML	Voice Extensible Markup Language
VoIP	Voice over Internet Protocol
VV&T	Verification, Validation, and Testing
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WAP Forum	Former forum focused on browsing and device provisioning protocols, Now part of OMA
WBEM	Web-Based Enterprise Management
W-CDMA	Wideband Code Division Multiple Access
Web	Reference to the World Wide Web
WG	Working Group

Wi-Fi	A term developed by the Wi-Fi Alliance to describe WLAN (WLAN) products that are based on the IEEE 802.11x standards.
Wi-Max	Worldwide Interoperability for Microwave Access
WS-*	Web services specification prefix
WSA	Web Services Architecture
WS-DD	OASIS WS Discovery and WS Devices Profile
WSDL	Web Service Definition Language
WS-I	Web Services Interoperability Organization
WS-SX	Oasis Web Services Secure Exchange
WTSC	Wireless Technologies and Systems Committee (ATIS)
XACML	Extensible Access Control Markup Language, sometimes EACML
XAM	eXtensible Access Method
XCAP	XML Configuration Access Protocol
XDM	XML Document Management
XForms	Extensible Markup Language Forms
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

APPENDIX C OSA/PARLAY SERVICE CAPABILITIES

Functionality	Parlay X - Service Description
Common – Part 1	Defines the common capabilities such as name spaces data definitions, fault definitions and WSDL descriptions of the interfaces
Third-Party Call – Part 2	For creating and managing a call created by an application. Allows set up of a connection between two parties from within the application
Call Notification– Part 3	Defines how a call should be treated for network initiated calls
SMS – Part 4	Allows applications to invoke SMS functions from within application
MMS – Part 5	Allows applications to invoke Multimedia Messaging that can map to SMS, EMS, MMS, IM, E-mail (*sendMMS*, *sendEMS*, *sendSMS*, etc).
Payment – Part 6	Payment for any content. Payment supports payment reservation, prepaid and post-paid payment
Account Management -- Part 7	Defines for pre-paid subscribers support for account querying, direct recharging and recharging through vouchers.
Terminal Status- Part 8	Provides access to the status of a terminal and allows notification of terminal status change. Terminal status is: Reachable, Un-reachable, Busy
Terminal Location – Part 9	Provides access to location of terminal plus notification of change of location or periodic change of location. Location is Latitude, Longitude and accuracy
Call Handling – Part 10	Allows accepting calls, declining calls, blocking calls forwarding calls and playing audio announcement
Audio Call – Part 11	Allows a way to provide voice message delivery
Multimedia Conference – Part 12	Allows creation of a multimedia conference and dynamic management of the participants and media involved in the call
Address List management – Part 13	Allows management of groups with support for group creation, query, access right management.
Presence – Part 14	The presence service allows for presence information to be obtained about one or more users and to register presence for the same
Message Broadcast -- Part 15	Message broadcast is a functionality that allows an application to send messages to all the fixed or mobile terminals in a specified geographical area.

Functionality	Parlay X - Service Description
Geo coding Part 16	Geo coding Web Service allow is the service developer to work with actual location addresses
Application Driven Quality Of Service – Part 17	Application Driven QOS is a service which enables applications to dynamically change the quality of service (e.g. bandwidth) available on end user network connections. Applications can register with the service for notifications about network events that affect the quality of service temporarily configured on the end user's connection
Device Management – Part 18	The Parlay X Device Management Web Service will allow applications to get information about device capabilities and push device configuration to a device
Multimedia Streaming Control – Part 19	The service provided to an end-user is consumption of streaming Multimedia.
Multimedia Multicast Control – Part 20	Allows for a third party (e.g. application) to control a multicast session, its members and multimedia stream, and obtain channel presence information
Content Management – Part 21	The content management web service enables uploading content into the network (or a third party content provider) and consuming content from the network (or a third party content provider)
Policy – Part 22	The Policy Web Service is defined to provide simple means for applications to make use policies to satisfy two purposes as follows: • The first one is to provide the user defined policies for the 3rd party applications who want to personalize their services by using their own preference expressed as policies at a high level. At this level, policies could be defined and managed by 3rd party applications, and applied to any policy enabled service. • The second one is to protect resources in network from unauthorized requests based on policies, therefore enables the network operators and service providers to control the access to their resources. Network resources can be accessed in a secure and controlled way and network operator could impose constraints on the usage of their services.

Table C-1: Parlay X Interfaces

Functionality	OSA Parlay APIs Description
Overview – Part 1	Contains the introduction, methodology and design paradigms used.
Common Data Definitions – Part 2	The generic data types used throughout the Parlay specifications.
Framework – Part 3	Describes how applications authenticate themselves to the network, how applications discover what facilities are available from the network, and fault and load management.
Call Control – Part 4	These APIs enable applications to set-up calls in the network, and include basic call control and multi-party call control functionality, among others.
User Interaction – Part 5	These APIs define how applications obtain information from the end-user, play announcements, send short messages, etc.
Mobility – Part 6	Mobility APIs enable applications to find the location of a terminal (handset) and enable applications to request notifications when terminals change location.
Terminal -- Part 7	This API enables an application to determine the capabilities of an end-user terminal (handset).
Data Session Control - Part 8	How applications manage data sessions initiated from terminals.
Account Management – Part 11	Enables applications to retrieve account information and transaction histories
Charging – Part 12	Controls how applications request payment for services ("content-based charging")
Policy management - Part 13	This API allows one to set-up policies and register for policy related events.
Availability management and presence – Part 14	These enable applications to obtain and set information about a user's presence and availability. For example, users may set specific presence information "I am at home", "I am at my desk" – similar to settings used in instant messaging applications.
Multi-media Messaging (MM) Service Capability Feature – Part 15	The Multi Media Messaging SCF (MMM SCF) is used by applications to send, store and receive messages either from within the context of a mailbox paradigm, or outside of it. MMM SCF also supports voice mail and electronic mail as the messaging mechanisms.
Service Brokering – Part 16	The Service Broker SCF enables the application to register its interest in particular traffic as part of service interactions

Table C-2: OSA Parlay Interfaces

APPENDIX D: POLICY USE CASES

Policy Use Cases removed for external transmittal.

APPENDIX E: SON-FG PARTICIPANTS

Chair		
BT	Matt Bross	Group Chief Technology Officer
Members		
Alcatel-Lucent	Gary Hanson	Strategy Director
Alcatel-Lucent	Ken Biholar	Director, North American Standards
Alcatel-Lucent	Omar Elloumi	
AT&T	Gary Munson	Technical Consultant- Strategic Standards
AT&T	Jenny Huang	Strategic Standards Manager
AT&T	Rich Erickson	Lead Member of Technical Staff
BT	Paul Muschamp	Head of IT Strategy
Cisco	Jay Gardner	Senior SE Manager
Cisco	Josh Merrill	Service Provider Consulting Architect
D&E Comm.	Troy Wingenroth	VP of Information Technology
Deutsche Telekom	Herbert Damker	Deutsche Telekom AG
Deutsche Telekom	Marcus Dormanns	Technology and Production Strategy, IT
Detecon	Thorsten Claus	Director
Ericsson	Asok Chatterjee	VP – Strategic Standardization, Public Affairs
GENBAND	Fred Kemmerer	Chief Technology Officer
Hewlett Packard	Gary Iosbaker	
Hewlett Packard	Gerry Winsor	Senior Architect
Hewlett Packard	Marc Brandt	
Huawei	Rick Townsend	Consultant
JDSU	Jay Stewart	Director of FMC/IMS Strategy
Juniper	Ravi Medikonda	Director of Service Provider Marketing
LG	Nandhu Nandhakumar	Senior VP, Advanced Technology
Motorola	Dean Skidmore	
Neustar	Tom Ely	Senior Director

Nortel	James McEachern	Strategist
Qwest	Rich Cerami	
Qwest	Andrew White	Director, NGN Architecture
Qwest	Michael Fargano	Industry Standards Program Manager
RIM	Suresh Chitturi	Standards Manager
RIM	John-Luc Bakker	Standards Manager
Sprint	Wing Lee	Director of Innovation Realization
Sprint	Erich Izdepski	Manager, Innovation Realization
TDS Telecom	Andy Sofronas	Enterprise Architect
TDS Telecom	Claudio Taglienti	Lead Architect
Tekelec	Gavin Cato	VP of Business Development
UTStarcom	Krishna Viswanadham	VP, Office of CTO
ATIS Staff		
	Tim Jeffries	VP Technology & Business Development
	Martha Ciske	Manager, Technology Programs & Research
	Paige Labourdette	Program Assistant, Technology Programs