A Framework for Abstracting Complexities in Service Delivery Platforms

Rolan Christian

A thesis submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy.

Johannesburg, July 2009

Declaration

I declare that this thesis is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Doctor of Philosophy in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this 17^{th} day of July 2009

Rolan Christian.

Abstract

The telecommunication (telco) and Information Technology (IT) industries are converging into a single highly competitive market, where service diversity is the critical success factor. To provide diverse services, the telco network operator must evolve the traditional voice service centric network into a generic service centric network. An appropriate, but incomplete, architecture for this purpose is the Service Delivery Platform (SDP). The SDP represents an IT-based system that simplifies access to telco capabilities using services. SDP services offer technology independent interfaces to external entities. The SDP has vendor-specific interpretations that mix standards-based and proprietary interfaces to satisfy specific requirements. In addition, SDP architectural representations are technology-specific. To be widely adopted the SDP must provide standardised interfaces. *This work contributes toward SDP standardisation by defining a technology independent and extendable architecture, called the SDP Framework.* To define the framework we first describe telecom-IT convergence and a strategy to manage infrastructure integration. Second, we provide background on the SDP and its current limitations. Third, we treat the SDP as a complex system and determine a viewpoint methodology to define its framework. Fourth, we apply viewpoints by extracting concepts and abstractions from various standard-based telecom and IT technologies: the Intelligent Network (IN), Telecommunication Information Networking Architecture (TINA), Parlay, enhanced Telecommunications Operations Map (eTOM), Service Oriented Architecture (SOA) and Internet Protocol Multimedia Subsystem (IMS). Fifth, by extending the concepts and abstractions we define the SDP framework. The framework is based on a generic business model and reference model. The business model shows relationships between SDP, telco and external entities using business relationships points. The reference model extends the business model by formalising relationships as reference points. Reference points expand into interfaces exposed by services. Applications orchestrate service functions via their interfaces. Service and application distribution is abstracted by middleware that operates across business model domains. Services, interfaces, applications and middleware are managed in Generic Service Oriented Architectures (GSOA). Multiple layered GSOAs structure the SDP framework. Last, we implement the SDP framework using standard-based technologies with open service interfaces. The implementation proves framework concepts, promotes SDP standardisation and identifies research areas.

Acknowledgements

The following research was performed under the auspices of the Center for Telecommunications Access and Services (CeTAS) at the University of the Witwatersrand, Johannesburg, South Africa. This center is funded by Telkom SA Limited, Vodacom, Siemens Telecommunications and the Department of Trade and Industrys THRIP programme. This financial support was much appreciated.

I thank God for my many blessings. I thank my role model and supervisor Prof. Hu Hanrahan for his support and guidance during the duration of this research project. I also thank my colleagues at CeTAS for their valuable inputs during this research. Last, I thank both my parents, brother and fiance for their love, support and patience.

Contents

List of Figures

List of Tables

List of Abbreviations

AAA	Authentication, Authorization, and Accounting
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BGCF	Border Gateway Control Function
BPEL	Business Process Execution Language
BSS	Business Support Systems
CAMEL	Customised Application for Mobile Enhanced Logic
CCF	Charging Collection Function
CMIP	Common Management Information Protocol
CORBA	Common Object Resource Broker Architecture
CRM	Customer Relation Management
CSCF	Call Session Control Functional
DPE	Distributed Processing Environment
DSL	Digital Subscriber Line
EAI	Enterprise Application Integration
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
eTOM	enhanced Telecommunication Operations Map
GGSN	Gateway GPRS Support Nodes
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GSOA	Generic Service Oriented Architecture
HLR	Home Location Registry
HSS	Home Subscriber Server
HTTP	Hyper Text Transfer Protocol
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IDL	Interface Definition Language
IMS	Internet Protocol Multimedia Subsystem
IM-SSF	IP Multimedia Service Switching Function

IN	Intelligent Network
IP	Internet Protocol
IPTV	Interactive Personalised Tele-Vision
IT	Information Technology
IVR	Interactive Voice Response Unit
JAIN	Java API for Integrated Network
JRE	Java Runtime Environment
MF	Mediation Function
MGCF	Media Gateway Control Function
MGW	Media Gateway
MPLS	Multiple Path Label Switching
MRFC	Media Resource Function Controller
MRFP	Media Resource Function Processor
NEF	Network Element Function
NGN	Next Generation Network
NGOSS	New Generation Operations Systems and Software
OCF	Online Charging Function
OMA	Open Mobile Alliance
OMG	Object Management Group
ORB	Object Resource Broker
OSF	Operations System Functions
OSS	Operational Support Systems
PDF	Policy Decision Function
PSTN	Public Switched Telephone Network
QAF	Q Adaptor Function
RTP	Real Time Protocol
SAG	Service Agreement Group
SCE	Service Creation Environment
SCF	Service Capability Feature
SCIM	Service Capability Interaction Manager
SCP	Service Control Point
SCS	Service Capability Server
SCP	Service Control Function
SDP	Service Delivery Platform
SGW	Signalling Gateway
SIBS	Service Independent Building Blocks
SID	Shared Information and Datal

SIP	Session Initiation Protocol
SME	Service Management Environment
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SS7	Signalling System Number 7
SSF	Service Switching Function
TCP	Transport Control Protocol
TDM	Time Division Multiplexor
TINA	Telecommunication Information Network Architecture
TMN	Telecommunication Management Network
TNA	Technology Neutral Architecture
TOM	Telecommunication Operations Map
UDDI	Universal Description, Discovery and Integration
UE	User Equipment
UML	Unified Modelling Language
UMTS	Universal Mobile Telephone System
VLR	Visiting Location Registry
WSDL	Web Service Description Language

Chapter 1

Introduction

Traditional *telecommunication company (telco)* networks provide basic, reliable, secure and quality voice communication between millions of customers. These networks include both fixed Public Switched Telephone Network (PSTN) and wireless Global System for Mobile Communications (GSM) cellular network. Overlay networks have enabled additional services, such as free-phone, voice-mail, messaging and pre-payment communication to be provided. Examples of overlay networks include the Signalling System Number 7 (SS7) [1] to provide signalling support and the Intelligent Network (IN) [2] to provide services. Each network adds its own complex mixture of hardware, software and communication protocols to the base telco network.

Other traditional networks include the enterprise and broadcasting networks. These networks use various technologies to deliver their services to their customers. For example:

- Enterprises use *Information Technology (IT)* based platforms to create applications and services. These platforms deliver services using a limited intelligence and unreliable packet-based transport, that is, the *Internet* or Intranets. Enterprise services are used by customers with intelligent devices. The platforms and devices manage end-to-end reliability, security and quality over the Internet.

- Broadcasters provide content-based services to customers using their own wired and wireless networks. Traditional broadcaster services are both unidirectional and bidirectional. However, these bidirectional services provide limited interactivity. In addition, some services provide limited customisation. In this work we do not focus on broadcast networks.

Historically, the telco, enterprise and broadcasting networks operate in separate domains. Each domain is administered by an authority. This authority regulates each network domain and ensures that each network operates within its domain and abides by various policies and rules.

Currently, each network is influenced by similar threats and opportunities in their respective domains. The threats and opportunities include the following:

- Transport innovations: new and improved systems enable interworking between various transport networks. These include interworking between the telco's circuit-mode PSTN and the Internet's packet-based transport network.

- Software innovations: various IT-based systems aid service development, deployment and management. In addition, these software systems provide mechanisms to support and simplify network and customer management.

- Customer requirements: customers demand basic and advanced services that support their professional and social lives. The services function independently of the customer device and location, but remains dependent on customer preferences.

- Advanced services: new services combine traditional voice with content, such as video or information. Services are customisable and adaptable to various stimuli, such as presence, availability, location and device capabilities. Also, advanced and legacy services are delivered over any network.

- Relaxed regulations: both technology dependent and independent factors influence regulative policies, such as transport networks and customer requirements. Hence, authorities define new or altered policies to break boundaries between various network domains. As a result, regulators aim to foster an open, fair, competitive and multi-service environment [3].

- Service-oriented environment: creation of a *Information and Communication Technology (ICT)* environment. The ICT environment represents the merging of various network domains into a single, competitive and service centric domain. Within the ICT environment operators integrate various information and communication-based technologies into their networks to support a variety of services.

The telco aims to manage threats and opportunities by adapting its network to provide advanced customer services and deliver them across heterogenous transport networks. Therefore, telco network operators are embracing the current evolutionary trend of *Convergence*.

1.1 Convergence

Convergence manifests itself in many ways. In this work, we focus on a particular aspect of convergence that is *telecom-IT convergence*. This form of convergence integrates traditionally separate telco and IT-based networks to provide a multi-service platform, that supports the creation, deployment, management and delivery of communication, information and content-based services across diverse transport networks. These services provide single to multimedia-based functionality, such as normal telephony, information browsing and interactive television. Thus, integrated telco and IT-based networks are able to benefit within the ICT environment, by providing plentiful services across varied technologies to satisfy customer needs.

Telecom-IT convergence is a *complex* undertaking since each network constitutes mixtures of systems that must interoperate. These systems are standard-based or proprietary and are dependent on specific technologies. As a result, various forms of convergence exist between telco and IT networks. By extending the convergence model of [4] we list the following forms of telecom-IT convergence. Also, these forms of telecom-IT convergence are summarised in *Table 1.1*.

- Device convergence: customers use a single device on various access networks, to consume multiple types of services.
- Transport convergence: interworking circuit mode networks and packet-based networks, including access and edge networks. Network interworking is managed by a multi-service network that is able to deliver any services across any interworked network.
- Functional convergence: diverse network functions interwork to support service delivery and management, such as billing.
- Service convergence: diverse network functions are abstracted and combined into reusable network services. These services provide application developers with open, managed, secure and technology neutral interfaces to network capabilities.
- Application convergence: integrate various types of application functionality and content to provide services to customers. Also, applications use network capabilities by accessing and combining network service interfaces.
- Content convergence: represents the managed collection of content providers, who provision their content using network capabilities. In addition, the network treats content similarly to a resource that is accessible to applications via network services, network functions, transport networks and customer devices.

	Telecoms	IT	Broadcasting
Applications and Content	Applications enhance telecom services. Seen in the IN.	Applications enhanced by communications.	Applications providing value added access to content.
	Converged applications provide a diverse range of services, that provide either voice, information, content or integrated forms of media.		
Services	Telecom type services are real-time and voice-based. Provides limited data services such as messaging.	Internet type services such as e-mail and web browsing. Both real-time and delayed but information rich.	Generic services used for accessing and consuming content.
	Converged services represent abstract forms of network functionality. Integrating multiple services creates converged applications.		
Functions	Supports delivery of telecom services to customers on telco networks. Supports billing and network interworking.	Provides limited intelligence that is used to deliver services across transport networks to customer terminals.	Provides needed functionality to encode/decode content for delivery across transport networks to customers.
	Converged functions provide a stable base that enables applications and services to operate over varied network infrastructure. Also, ensures service delivery across varied transport networks.		
Infrastructure	Circuit mode and packet access and core networks.	Interworking of multiple packet-based networks.	Varied networks.
	Converged infrastructures represent a collection of diverse networks that are integrated via a multi-service network. Includes capabilities of customer device.		

Table 1.1: Summary of telecom-IT Convergence

Already, enterprise networks are evolving to manage some forms of convergence. For example, enterprises are providing Internet-based services that combine multiple forms of functionality and media, such as multiparty voice communication, short messaging, video streaming and information browsing. These Internet services are increasing in usage and popularity, by both fixed and mobile Internet and telco customers.

To respond to Internet-based services, the telco may promote its own *"Internet-like"* services that are enhanced by using telco network capabilities. For example, these telco services operate over reliable, secure and quality transport networks. In addition, telco services may provide added functionality based on fixed and mobile customer terminals. Hence, these Internet-like services provide the telco with an alternate revenue stream to the constant voice service revenues [5].

The current stage of telco evolution includes telecom-IT convergence. Hence, the evolved telco network must enable content and service delivery using any or all underlying functions, transport networks and devices. However, underlying network complexities must be hidden from the application providing the customer service. As a result, the telco network must become *service centric*.

1.2 Service Centric Networks

We use the *concept* of a service centric network to describe a network that provides an abundance of services to its customers, independent of the underlying network details. These details include diverse interworking transport networks, varied customer terminals, transmission of protocols, distribution of network capabilities and resources and varied network service implementations.

Services provided by service centric networks include basic services, such as voice, and advanced services, such as real-time television streaming. Service centric networks provide an environment for any service to be created, maintained and managed. In addition to services, these networks also manage content needed by services.

To support service creation, service centric networks *abstract* the underlying complexities of converged networks, such as integrated transport networks, functions and services. In addition, service centric networks abstract customer domain complexities, such as devices and location. These abstractions are represented as generic and reusable software-based *services* that are incorporated into applications. It is these applications that provide basic and advanced services to customers.

A variety of service centric network *architectures* exist. We describe three specific architectures, the Next Generation Network (NGN) [6], Service Oriented Architecture (SOA) [7] and Service Delivery Platform (SDP) [8].

1.2.1 Next Generation Network

The NGN *concept* is defined by [9] as a packet-based network providing a variety of services, such as telco and Internet services, using various quality of service enabled transport technologies and service-oriented functions that are abstractions of underlying transport technologies [6].

Based on the above definition, the NGN aims to support the evolution of the telco circuit switched networks into a packet-based all purpose service centric network. In addition, the NGN concept guides integration of telco networks with and other network types.

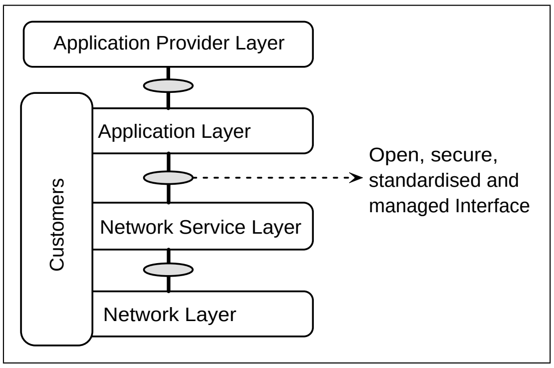


Figure 1.1: An Abstract NGN Representation

Adapting the NGN definition, the NGN exhibits the following characteristics:

1. provides a quality assured *packet-based* transport that interworks with other networks, such as the PSTN and Internet;

2. delivers telco and Internet-based *services* to fixed and mobile customers over its packet transport;

3. *separates* network independent service related functions from underlying transport technologies;

4. offers controlled *open access* to network service functions to both customers and application providers; and

5. facilitates the provision of services from application *providers* to customers.

Based on these characteristics, the NGN supports most forms of convergence.

The NGN breaks with traditionally closed telco network architectures and business models. By providing managed open access, the NGN enables the telco to partner with diverse entities, such as application developers or other network operators. Application developers may be external IT-using enterprises that create services for telco customers. Thus, opening of the telco network increases the number of connections the network has and therefore the network's value increases [10].

A NGN architecture is shown in *Figure 1.1*. In the figure, the application provider layer contains external IT-based infrastructure used by telco partners to create and support applications. The application layer houses applications developed by external IT-using enterprises. These applications invoke telco network resources and capabilities via network service interfaces. The network service layer contains a diverse range of network services that expose access to telco network capabilities via their technology neutral interfaces. The network layer houses the physical equipment that constitutes the telco network.

The NGN architecture also shows a customer domain that intersects the application, network service and network layers. As a result, each layer contributes applications, network services and equipment to the customer domain. This enables customers to access the NGN and various telco or Internet-based services.

The NGN representation in *Figure 1.1* is highly abstract and does not dictate technology choices for its implementation. Although, network operators must consider using standardised technologies to implement each NGN layer to ensure interworking with other NGNs. An example technology that can be used to implement parts of the NGN is the SOA.

1.2.2 Service Orientated Architecture

Enterprises manage convergence by focusing on integration of IT systems, rather than developing new systems [11]. These IT systems provide the needed functions to satisfy business processes. In addition, IT systems may belong to the enterprise or external partners. These IT systems are technology and distributed dependent. The Service Orientated Architecture (SOA) is therefore defined to ease integration of diverse and distributed systems.

The SOA represents an architecture and set of *standardised* technologies that enable enterprise resources to be used on demand and exposed to customers and external partners [12]. The SOA fulfills these properties by managing a collection of reusable and technology neutral services [13]. These services are called *web services* and they each *abstract* enterprise resources, such as applications, databases, devices, transport networks and other web services.

The collection of web services represents a service repository. Services, within the repository, are reusable in multiple application development efforts. Thus, web service functionality is *orchestrated* into applications that automate business processes.

A SOA representation is shown in *Figure 1.2*. The figure illustrates various entities, such as web services, applications and a service registry. The SOA entities may be programmed using different technologies. But, to ensure interoperability between the communicating entities, standards-based SOA technologies are used. These standardised technologies include the following:

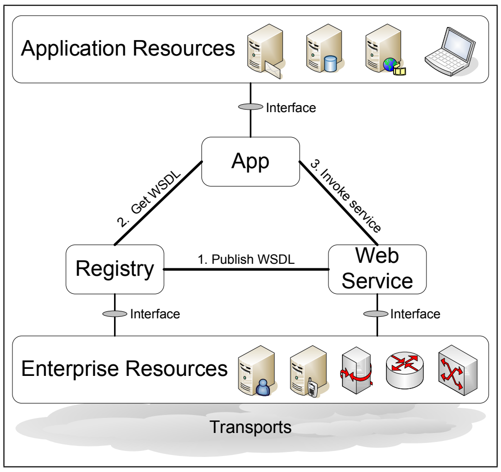


Figure 1.2: A SOA Representation

- Web Service Description Language (WSDL) [14]: describes the location and functionality provided by the web service.
- Universal description, discovery and integration (UDDI) [15]: provides the mechanisms required for applications to find web service WSDLs.
- Simple Object Access Protocol (SOAP) [16]: enables technology-neutral communication between UDDI databases, applications and web services.
- Hyper Text Transfer Protocol(HTTP) [17]: provides the means to transport SOAP across networks.

In the figure, web services publish their description in WSDL to the service registry. Obtaining the web service description occurs using one of two methods. First, during application development the web service description is discovered from the registry. Using the description, developers incorporate the web service functions into the application implementation. Second, during run-time applications dynamically discover the web service description and invoke the corresponding web service functions. All discovery and invocation communication occurs using SOAP and HTTP.

The technologies used for the SOA are standardised. However, the SOA is locked into using these Internet-based technologies only. Also, some of these technologies are not fully *mature*. Therefore, the SOA is not an adequate candidate for evolving the telco network into a service-centric network, that manages the various forms of telecom-IT convergence. However, the telco network may reuse SOA concepts to offer generic telco services to enterprises. The enterprises may use these services to create *"telecoms-enhanced"* applications for their own customers or telco customers.

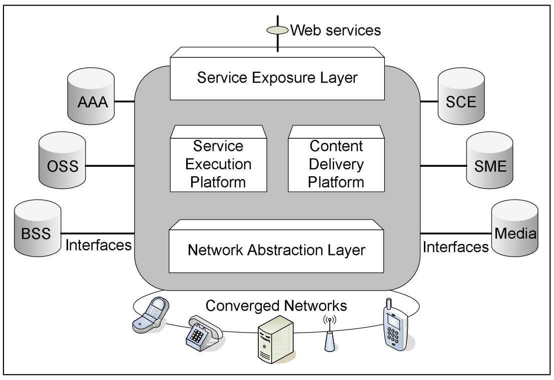


Figure 1.3: Proposed SDP Architecture

Another architecture that builds upon both NGN and SOA is the SDP.

1.2.3 Service Delivery Platform

The Service Delivery Platform (SDP) is designed specifically for delivering services to users of communications technologies [18]. The SDP uses telecoms, IT or content-based capabilities to deliver services to customers, independent of network technologies. These services include number translation, click-to-dial, location based services, virtual call centres and IT type services, such as mobile gaming [8].

A proposed SDP architecture that is adapted from [8] is illustrated in *Figure 1.3*. This SDP representation is based on common vendor product functions and constitutes functional abstractions, that simplify complexities within a telco network. These abstractions and their functions are:

1. Service Creation Environment (SCE) and Management Environment (SME): is used by internal application developers to create applications that access telecom capabilities independent of network technologies.
2. Service Exposure Layer: provides external application developers with access to generic and reusable services that abstract applications contained in lower platforms. As an example, this layer may be SOA-based and contain web services.
3. Service Execution Platform: hosts various applications that provide telco, Internet and content-based services to customers.
4. Content Delivery Platform: manages the provision and delivery of content to customers using content-based services. Content is stored in media repositories that are accessible via interfaces.

5. Network Abstraction Layer: provides a technology-neutral set of capabilities that enable the delivery and management of services across converged networks.

6. Management Platforms: integrate into all layers and platforms, so as to administer the SDP. Each SDP abstraction has access to management resources and capabilities, such as the *Operational and Business Support Systems (OSS/BSS)*.

Based on the definition and proposed architecture, the SDP aims to provide a complete managed *environment* that offers various service abstractions to external application developers. These abstractions simplify underlying telecom systems and technologies. However, the proposed SDP representation has disadvantages:

1. Choosing implementation *technologies* for each SDP abstraction is difficult. The interfaces must be standardised so as to promote SDP interworking. Also, the technologies must provide relevant abstractions and support SDP management.

2. *Customer-related* abstractions are not defined. In addition, the platforms or layers do not simplify customer related activities. For instance, access, service provisioning and customer to service signalling is not abstracted.

3. The architecture provides a single abstracted *view* on the SDP. Missing SDP views include business and domain models, functional groupings, information distributions and application logic relationships. The missing views must provide more abstractions that are structured using specific architectures.

4. Access to telecom management facilities is provided by proprietary interfaces. In addition, how the SDP itself is managed is not defined. Similar to points 1 and 3, a management architecture is required that is implementable using technologies with standard-based interfaces.

As a result of these limitations, network operators *build or buy* custom SDP solutions that enable them to create limited services. These SDP implementations are diverse and follow no agreed or *standardised* interpretation. This inevitably leads to SDP solutions being dependent on various technologies that are standards-based and *proprietary*. This effects the openness of the SDP and interoperability between other SDP solutions. Therefore, *solving* these and other SDP limitations is the focus of this research.

1.3 Problem Statement, Aim and Objectives

The convergence trend produces various telecoms and IT service-centric network architectures, such as the NGN, SOA and SDP. These architectures may be used to evolve telco voice-centric networks into service-centric networks. Each architecture provides varying abstractions to hide convergence and network complexities. We consider the SDP the most generalised approach for evolving the telco network into a service-centric network and to benefit from convergence. However, the SDP is not standardised.

The SDP concept demands abstractions to simplify various complexities. These abstractions must be structured and their implementations standards-based. Hence, this research responds to the question: *what are the convergence and network complexities that can be simplified by the SDP and what architectures, abstractions and interfaces must be developed to aid this simplification?* Based on this problem statement we pose the following sub-questions:

1. Who are the external IT-using enterprises that benefit from the development of a standard-based SDP?
2. What is the SDP business model used to manage the business relationships between telco and external IT-using enterprises?
3. How does the SDP enforce standard-based interactions between the telco and external IT-using enterprises to satisfy business relationships?
4. What are the services that the SDP defines to abstract telco network resources and capabilities?
5. What are the service interfaces that the SDP must expose to external IT-using enterprises?
6. How must the architecture in *Figure 1.3* be elaborated or simplified to promote a standard-based SDP.
7. What design pattern do we use to structure the SDP architecture such that it is technology neutral and extendable?
8. How do we incorporate OSS/BSS into the SDP architecture such that we promote a standard-based SCE and SME?

Our research aims to answer these questions and as a result define a *SDP framework* that contains various levels of abstraction. The framework and its abstractions simplify various complexities and guide telco network evolution into a SDP. The objective of the framework is to also promote SDP *standardisation* where possible using established standards. This is achieved by applying the framework to create technology, distribution and implementation independent SDP architectures. The generic architectures motivate the use of standard-based *technologies* to implement the SDP. By using standard-based technologies, the SDP becomes truly open to diverse IT using enterprises and their applications. Also, the SDP can access and use converged network capabilities, including customer devices. Hence, being standard-based ensures interoperability between SDP implementations.

We aim to prove the concept of the SDP framework by developing a *SDP architecture* and implementing some of its parts using *standard-based* technologies.

1.4 Outline of Thesis

Telecom-IT convergence has implications on both telco and IT-based networks. A major implication is the interworking between telco and IT systems. This interworking requires new converged systems to be developed. These new systems are complex since they must perform both telecom and IT-based functions. One such system is the SDP. A methodology is used to develop complex systems by defining their architectures using various viewpoints. In chapters 2 and 3 we define telecom-IT convergence and a viewpoint methodology.

- Chapter 2: we describe telecom-IT convergence as a complex process. We define the process and discuss its implication on both telco and IT-based networks. In addition, we define a strategy to manage telecom-IT convergence.
- Chapter 3: we discuss complex systems, their development and their relation to defining the SDP framework. We elaborate on reusable tools that manage complexity and contribute to the definition of the SDP framework. In addition, we explain the methodology used to define the SDP framework.

Various legacy and current standards in telecoms and IT are useful in conceptualising and defining the SDP. These standards contribute towards uncovering SDP requirements, abstractions for the framework and reusable interfaces to implement the framework. As a result, each of these standards provide a specific viewpoint on the SDP and the development of its framework. In chapters 4 to 8 we examine such standards.

- Chapter 4: we define a perspective on the SDP by using legacy Intelligent Network (IN) [2] and Telecommunication Information Network Architecture (TINA) [19] standards.

- Chapter 5: we define a perspective on the SDP by using the Parlay [20] service platform standards.
- Chapter 6: we define a perspective on the SDP by using management framework standards, such as Telecommunication Management Network (TMN) [21], Telecommunication Operations Map (TOM) [22] and enhanced Telecommunication Operations Map (eTOM) [23].
- Chapter 7: we define a perspective on the SDP by using the enterprise SOA standard.
- Chapter 8: we define a perspective on the SDP by using the Internet Protocol Multimedia Subsystem (IMS) [24] standard.

Results from the viewpoints on the SDP include various abstractions. These abstractions represent the building blocks that structure the SDP framework. In addition, these building blocks are technology neutral and may be implemented using any suitable telecoms and IT-based technologies. Chapters 9 and 10 represent the synthesis of these abstractions into the SDP framework and a proof of concept implementation.

- Chapter 9: we define the SDP framework by integrating concepts obtained from all perspectives. The framework is defined using stages containing abstraction and architectures. We also compare some SDP interpretations to the framework.
- Chapter 10: we prove the concepts of the SDP framework by defining a SDP architecture and implementing its parts. For the implementation we use service interfaces provided by various standards-based technologies.
- Chapter 11: we discuss the contribution of the framework to SDP standardisation. In addition, we summarise our results and present recommendations for future work.

Chapter 2

Convergence

In this research we focus on telecom-IT network convergence. A definition of telecom-IT convergence is integrating reliable telco voice (and data) services with information (and media) capabilities supported by IT and the Internet [25]. This form of convergence is evident in the SDP architecture, shown in *Figure 1.3*. The SDP architecture incorporates both content delivery and telco services, and exposes these capabilities to enterprises using IT-based mechanisms. In addition, the SDP architecture uses network capabilities provided by converged telco, IT and Internet networks.

For the telco, telecom-IT convergence implies the provision of simple and converged services to customers over the telco network, IT-based enterprise networks and Internet. Simple services include normal telephony, messaging and information browsing, while converged services integrate combinations of voice, media and information. Telecom-IT convergence also implies the use of IT-based mechanisms by the telco to develop, deliver and manage services.

Convergence aims to allow telco and IT network infrastructures to interwork. We define infrastructure as the *arrangement, distribution and connection of hardware and software required to provide services*. However, heterogeneous telco and IT-based infrastructures hinder convergence, for example, circuit and packet-based transport networks for delivering services across telco networks and the Internet respectively. We interpret telecom-IT convergence as a *process* that allows interoperability between telco and IT-based infrastructures, such that numerous services are delivered to customers.

2.1 The Process of Separation and Integration

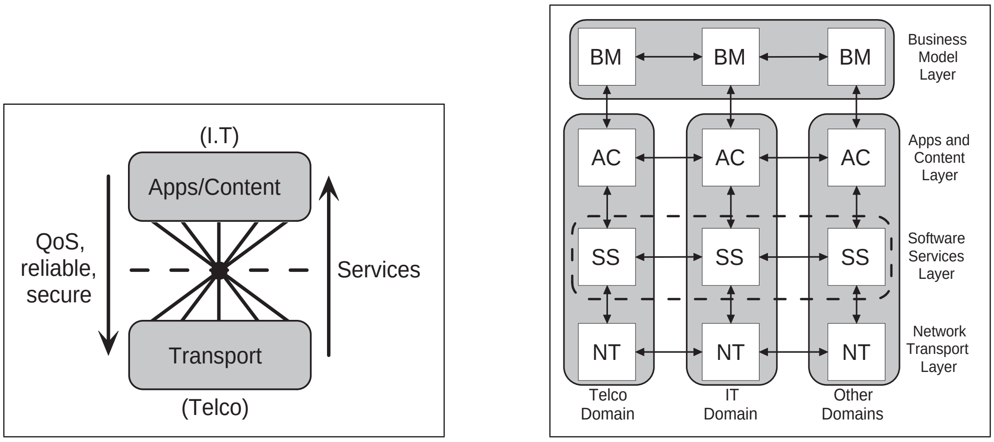
Telco networks contain a diverse infrastructure. Each infrastructure contains various levels of functionality, such as network services, network functions and transport networks. Traditionally telco infrastructure is separated into *vertical* silos. Each silo represents an independent system that provides limited services by using specific network capabilities. In addition, a silo operates on a particular transport network. Examples of silos include both legacy fixed and mobile telco PSTN/IN service platforms.

The vertical nature of telco network infrastructure is caused by environmental factors, such as business drivers, regulations, technologies and customer requirements. However, as environmental factors change, the telco network evolves. The current evolution aims to support interworking between telco network infrastructure and IT-based infrastructure, that is, telecom-IT convergence. To provide infrastructure interworking, the telco decomposes its silos by *horizontally separating* their common functions into layers.

Infrastructure functions are decoupled from their physical representation into abstract entities. As a result, the complexity of the infrastructure is decomposed into manageable and reusable functional entities. In addition, functional entities focus on particular areas of the network infrastructure. For example, *Figure 1.3* implies the use of layers to horizontally separate network abstraction and service exposure functionality. These sets of functionality abstract access to network capabilities, services and content. The figure also uses a platform layer to separate service execution and content delivery functionality. This functionality abstracts access to telco services and content used by services.

Within a layer, functions enable *horizontal integration* of telco and IT-based infrastructure. For example, *Figure 1.3* shows the integration of platforms with OSS/BSS and content repositories, via interfaces. Also, the figure shows converged networks that contain integrated telco transport networks, IT-based transport networks and Internet. The integration of these networks is achieved by horizontally interworking circuit mode and packet-based transport networks.

Layers also enable *vertical integration* of telco and IT-based infrastructure, by using their functions as *points of integration*. For example, *Figure 1.3* uses the exposure layer as a point of integration between IT-based applications and telco network capabilities. Also, the network abstraction layer is a point of integration between applications/content and converged networks. The platform layer provides the interworking between service exposure and network abstraction layers.



(a) Separation and Integration

(b) Horizontal and Vertical Convergence

Figure 2.1: Convergence of Infrastructure

A representation of infrastructure separation and integration, adapted from [26], is illustrated in *Figure 2.1(a)*. The figure depicts converged infrastructure that is horizontally separated into reliable, secure and quality of service guaranteed transport networks, that expose functionality using abstract *network services*. The infrastructure is also horizontally separated into application and content delivery functions. These functions vertically integrate with the transport networks by using network services. The figure applies to the convergence of enterprise IT solutions with telco networks.

By applying the separation and integration process to telco and IT infrastructures we present *Figure 2.1(b)*. The figure is derived from [26, 27]. The figure depicts three vertically separated network domains: telco, IT and other (may include broadcaster networks). Across each domain, common independent functionality is separated into horizontal layers. These layers being applications/content, software-based services and transport networks. The figure shows horizontal integration across layers. Also, the figure uses the services layer as a point of integration between layers.

As a result of applying both forms of integration, *Figure 2.1(b)* illustrates two types of telecom-IT convergence:

- Vertical Convergence: represents vertical integration between layers within the same domain. For example, the telco integrates its *internal* legacy systems and new systems, such as integrating PSTN and IN platforms with new service platforms, content stores, management facilities and high-speed transport networks.

- Horizontal Convergence: represents horizontal integration within layers, across different domains. For example, telco voice applications integrate with *external* content-based applications provided by IT using enterprises. Also, telco services may integrate with services located in external enterprises. This form of convergence also applies to the integration of transport networks, such as the telco and Internet. As a result, integrated transport networks support the integration of applications, content and services across network domains.

In *Figure 2.1(b)*, business models are horizontally integrated. This integration illustrates the convergence of business activities between telco and IT networks. The main activity focuses on providing services to customers. Hence, telco and IT business models are joining, such that service development, deployment and delivery is shared across multiple network domains. The figure does not show management applications and services, but they are contained in each domain and layer. These applications and services may also integrate vertically and horizontally.

Both horizontal and vertical convergence are realised by interworking networks' software and transport technologies. For example, integrating different signalling protocols, switching technologies, application implementations and content. However, this integration has various implications on the telco network.

2.2 Implications of Convergence

Telecom-IT convergence has the following implications:

- Telco and IT business models must integrate into value chains, with both benefiting from the development and delivery of services to customers.
- Circuit and packet switching must interoperate, to reliably transmit any form of data.
- Signaling protocols must interwork across networks to support service delivery.
- Service platforms must operate independent of the underlying network technologies and provide functionality to diverse application developers.
- Application development technologies must provide tools to support application creation, independent of the service platform that applications access.
- Management systems, such as OSS/BSS, must administer mixtures of terminals, transport networks, services and application technologies, across diverse networks.

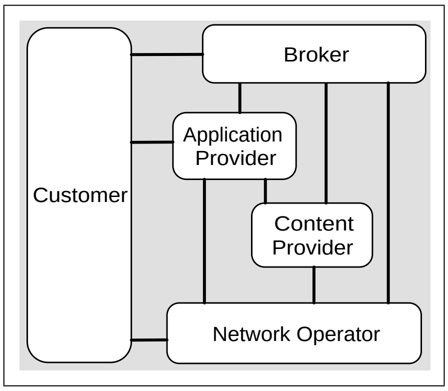


Figure 2.2: Example of a Converged Networks Business Model

We describe each of these convergence implications in the following sections.

2.2.1 Business Models

Offering a diverse range of services is essential within the ICT environment. To support service development the telco opens the network to a variety of external 3$^{\text{rd}}$ party *enterprises*. Examples of these enterprises include service providers, application developers or content providers. With this opening, telco and enterprise *partnerships* are formed. The aim of these partnerships is to support the creation and delivery of a variety of services to customers. As a result, new and innovative business models are established. An example of a converged business model is shown in *Figure 2.2*.

The business model depicts the following entities:

1. Network Operator: provides connectivity and access to its network resources and capabilities to other entities in the business model.
2. Application Providers: invoke common network functions to enhance applications that provide *value added* services to customers. These services include basic voice, video streaming, multimedia conference and mobile gaming.
3. Content Providers: manages, sells and delivers content needed by application developers. Two categories of content exist. First, traditional content such as voice announcements used by an Interactive Voice Response Unit (IVR). Second, new content needed by specialised services, such as images, audio, video, music, multimedia or information.
4. Brokers: enable interactions between business entities. For instance, a broker retails

services from application providers to customers. Also, brokers retail content from content providers to application providers.

5. Customers: are individuals or enterprises that subscribe, consume and pay for services.

With this model a value chain is created, where all partners benefit. For instance, application providers access abundant network functions to create *"telecoms-enabled"* applications. Also, content providers access network functions to delivery content to a variety of consumers. Via brokers, application providers access diverse content managed by content providers. Using the telco network and brokers, application providers sell their services to a large customer market.

Based on the above business model and scenarios, brokers are key to a converged business model [28]. Brokers provide a single point of access between multiple customers, application providers and content providers. Brokers therefore further promote *interactions* between entities in the converged business model. As a result, brokers contribute to service delivery. The telco network operator benefits from the increased activity in its network.

The business model benefits internal and external application development by providing access to connectivity and network functionality. This functionality is *reusable* and reduces time and cost in both internal and external application development [18]. Also, integrating more partners into the business model ultimately connects more entities to the network, thereby increasing the network's value [10].

The business model is *generic*. By remaining generic the business model entities may be *decomposed* into additional business partners that participate in specific business roles. For instance, application providers may be decomposed into service providers that sell services to customers. These service providers act on behalf of application providers, who only create the service. The business model is also *dynamic* and can easily evolve when influenced by the convergence process.

We define various business models for the SDP in *Chapters 4 , 5, 6* and *7*.

2.2.2 Network Technologies

Categories of emerging network technologies include user and network transmission methods, network elements, service, application and management protocols and platforms. Examples of network technologies are:

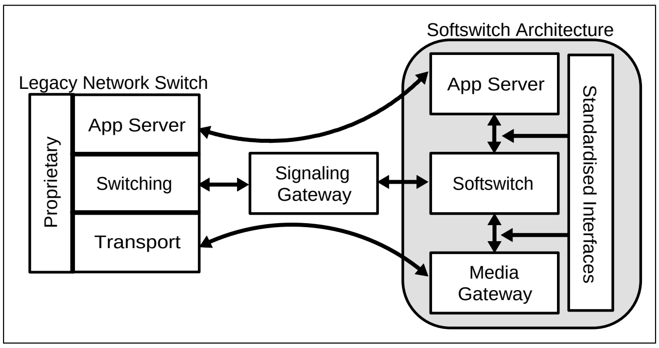


Figure 2.3: Evolution to a Softswitch Architecture

1. Customer Access: increased bandwidth provided by fixed and wireless mechanisms, such as Digital Subscriber Line (DSL) [29] and Universal Mobile Telephone System (UMTS) [30]. Technologies like these enable complex services to be offered and delivered over the network to users with intelligent devices [31].

2. Transport Switching: includes legacy circuit-orientated Time Division Multiplexing (TDM) switching. Also, virtual circuit packet-orientated Asynchronous Transfer Mode (ATM) [32] and Multiple Path Label Switching (MPLS) [33] are used to transport various types of data including protocols.

3. Transport and Signalling Protocols: the Internet Protocol (IP) [34] is used to transport various forms of data across both telco and IT networks. The IP also transports signalling protocols such as Transport Control Protocol (TCP) [35], Real Time Protocol (RTP) [36] and Session Initiation Protocol (SIP) [37] across networks. These signalling protocols are used to support different services on both telco and IT networks, such as voice, web browsing and video streaming.

Convergence requires the *abstraction* of complex, heterogenous and integrated network technologies, to enable service development. As a result, the *softswitch* [38] is defined. The softswitch enables different signalling and switching technologies to interoperate and reliably transmit any form of data. Therefore, the softswitch delivers services across different networks. Also, the softswitch enables application development, independent of network technologies.

The softswitch architecture adapted from [39] is shown in *Figure 2.3*. The architecture illustrates evolution from proprietary closed switching environments to the softswitch promoting open and *standardised* interfaces to network functionality. The softswitch architecture represents a functional decomposition of the network into the following functional abstractions:

- Media Gateway: is controlled by the softswitch and manages connections between different transport networks, such as telco data networks, PSTN and the Internet.
- Signaling Gateway: translates protocol messages originating from one network into other network specific protocol messages and transports them to the softswitch.
- Softswitch: contains logic for signaling, call processing and controls the media gateway. Also, it provides application servers with open, standardised and abstract interfaces to network functions.
- Application Server: houses applications that uses the softswitch interface to access network functionality. These functions enable service delivery to customers on various networks. An application server may be administered outside the network by an external enterprise, such as an application provider.

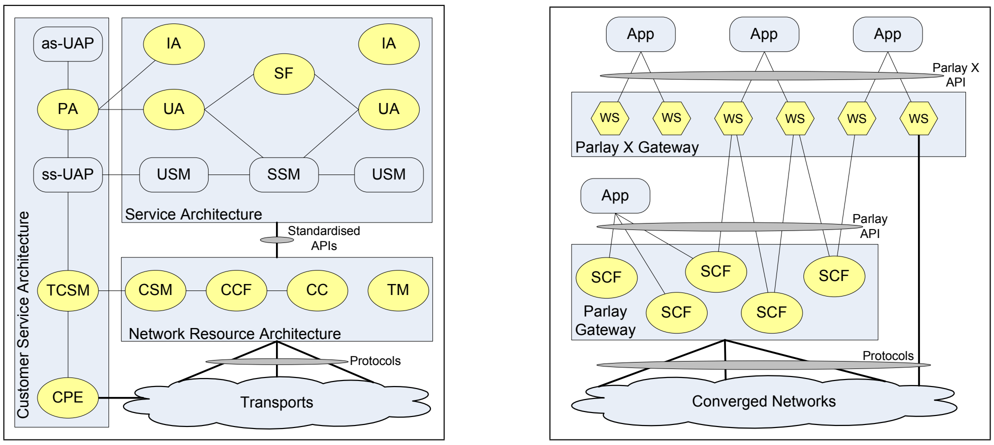
Softswitch specific protocols are used to communicate between the softswitch abstractions. Examples of these protocols include the set of Signaling Transport (SIGTRAN) [40] protocols and the Media Gateway Control (MEGACO) [41] protocol.

The softswitch and associated protocols provide the basis for other forms of networks to be created, such as the Internet Protocol Multimedia Subsystem (IMS) [24]. The IMS reuses softswitch functional abstractions, but decomposes them further and integrates Internet protocols into its packet-based network. The Internet protocols include SIP and Diameter [42]. We discuss IMS concepts that are applicable to the SDP framework in *Chapter 8*.

The softswitch architecture provides a foundation of *concepts* that aid telecom-IT convergence. These concepts and principles include decomposition of the telco network into various high level functional abstractions. These abstractions collectively represent a new open, secure and standardised network environment. These concepts also support the converged business model, since it promotes network openness to external enterprises. In addition, the abstractions are technology independent, such that they hide network integration and simplify service development and delivery.

2.2.3 Service Platforms

Traditionally, service platforms are specific to the networks they work on, such as the IN and Customised Application for Mobile Enhanced Logic (CAMEL) [43] for the PSTN and GSM respectively. By contrast, convergence requires the separation of service platforms from the underlying network specifics. Also, service platforms must promote reusability of network capabilities and resources in service development, but remain network *neutral*.



(a) TINA Service and Network Architectures (b) Parlay and Parlay X Architectures

Figure 2.4: Service Platform Architectures

The first form of network abstraction in service platforms are evident in the *planes* of the IN conceptual model. Though the IN provides abstraction, they are limited. As a result, IN application developers must be knowledgable in telecoms call states and call processing. Service platforms, based on extended IN concepts, aim to ease service development independent of underlying network complexities. The most influential in promoting thinking on service platforms are the legacy Telecommunication Information Networking Architecture (TINA) [19] and current Parlay [20] service platforms.

The TINA standards define a service architecture [44] and network resource architecture [45]. Both architectures are *complete and rich* with principles, concepts, computational objets and associated data structures. For instance, the architectures promote the separation of service and network related functions into abstract and reusable objects. Examples of objects include a service session manager and communication session manager. These objects and their data structures are reused in service development, deployment and delivery. An overall representation of both TINA architectures are shown in *Figure 2.4(a)*. The figure does not show all computational objects.

The Parlay Group [20] defines two service platform architectures containing *Application Programming Interfaces (APIs)*, that promote opening of telco networks to external IT using enterprises. These architectures are Parlay and Parlay X. Parlay groups APIs into Service Capability Features (SCF), while Parlay X uses web services [7]. Each API defines objects and data structures to describe how authorised applications use network functions to deliver voice, information or multimedia services to customers on converged networks. Also, APIs abstract network complexity, such as transport networks and protocols, for application developers. An abstract representation of both architectures are shown in *Figure 2.4(b)*.

The TINA, Parlay and Parlay X standards provide a wealth of reusable principles, *concepts*, objects and data structures. For example, these standards abstract service and network complexities into architectural layers that constitute software-based objects. In addition, the layered abstractions simplify network complexities and promote external access to network capabilities and resources. All these reusable parts contribute to the definition of service platform architectures that support telecom-IT convergence.

We discuss IN, TINA, Parlay and Parlay X (including SOA) service platform concepts that are applicable to the SDP framework in *Chapters 4, 5* and *7*.

2.2.4 Application Environments

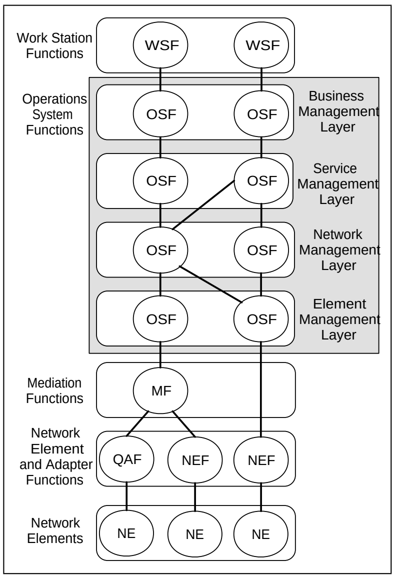
To benefit from convergence, telco service platforms must be accessible by a range of application development environments. Application providers typically use development environments such as .Net [46] or Java [47]. These technologies contain execution environments and provide standards-based APIs, that further abstract telco and IT network technologies. These technologies include service platforms, such as Parlay and Parlay X. Hence, these technologies provide software-based developer *tools* to support application development. However, the onus is on the application provider to determine the appropriate technologies needed to support their specific business, service and customer requirements and telco network provider.

2.2.5 Management Systems

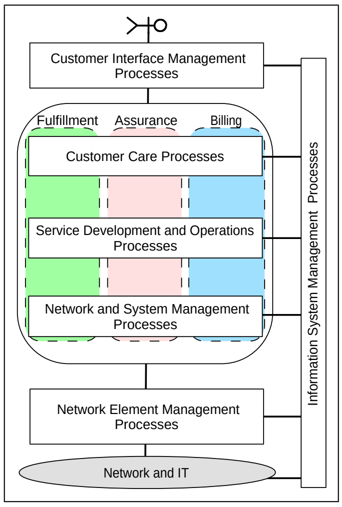
The telco must manage network and service activities, such as:

- business relationships and processes defined by a converged business model;
- physical equipment implementing platforms and transport networks;
- signalling systems that control transport networks and service delivery;
- services used in application development;
- functions used for accessing network capabilities; and
- services used by customers and internal users.

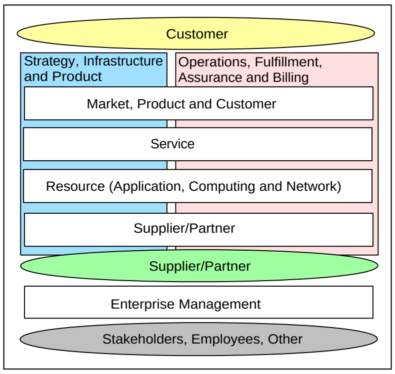
To aid telco management, various management frameworks may be implemented. These frameworks include the Telecommunications Management Network (TMN) [21], Telecommunications Operations Map (TOM) [22] and Enhanced Telecommunications Operations Map (eTOM) [23].



(a) TMN Layers and Functions



(b) TOM Layers and Domains



(c) eTOM Layers and domains

Figure 2.5: TMN Architecture, TOM Framework and eTOM Framework

TMN was intended to manage the legacy IN, SS7 and PSTN. Similar to the IN, the TMN is an overlay network that integrates with telco network elements. Each network element provides management *functions* that are executed during management events. Management functions are abstracted into simpler service and business oriented functions. As a result, TMN structures these functions into a layered architecture. When management events occur, management functions are executed across the architecture layers. Examples of management functions include mediation, operations, service and business functions. The TMN architecture is illustrated in *Figure 2.5(a)*.

TOM simplifies the complexity of implementing TMN and provides a framework that aids the development of OSS/BSS architectures and implementations. The TOM framework uses a customer and *service oriented* approach to telco management. In addition, TOM reuses the TMN layered and functional hierarchy approach. However, TOM defines various *business processes* within each layer. Each business process is activated by inputs and produces outputs, that are inputs for other processes. As a result, a chain of processes or process flows represent an *end-to-end* management activity [21]. Each business process is grouped into a process flow domain. Three domains exist: service fulfillment, service assurance or service billing. The TOM framework is illustrated in *Figure 2.5(b)*.

Currently, the eTOM framework is used to define OSS/BSS for the NGN, within the context of convergence. eTOM is derived from TMN and TOM. However, eTOM incorporates the role of the *Internet* in managing the telco, its services and customers. eTOM is defined in various stages, called process levels. Initial process levels define abstract layers and domains. These abstractions are then further decomposed into business processes. Additional decomposition produces more detail on business processes. An abstract representation of the eTOM framework is shown in *Figure 2.5(c)*.

The eTOM framework is used to build distributed IT-based systems that provide telco OSS/BSS. These systems are large and complex since they administer all parts of the telco network and enable Internet access to specific telco network management capabilities. Currently, the New Generation Operations Systems and Software (NGOSS) [48] initiative incorporates the eTOM framework into a guiding methodology to create telco OSS/BSS. The NGOSS methodology provides additional frameworks that identify shared information and technology neutral architectures for the OSS/BSS.

TMN, TOM and eTOM prescribe the *abstraction* of the telco network into simpler manageable parts. These parts are structured using architectural styles, such as layers and domains. However, TOM and eTOM identify business processes within their respective architectures. Business processes represent abstracted forms of management services that are provided by OSS/BSS platforms. In addition, some business processes are accessible by the customer and the telco partners. These frameworks and their processes contribute to the development of OSS/BSS platforms required to manage telecom-IT convergence.

We discuss TMN, TOM and eTOM concepts that are applicable to the SDP framework in *Chapter 6*.

2.3 SDP as a Strategy for Convergence

The convergence process and its resulting implications illustrate the *complexity* faced by telco network operators. This includes the formation of converged *value chains* and business relationships between partners. Also, *technology* choices that enable convergence are numerous and varied. Though some technologies are standardised and promote interoperability, each provide various levels of abstraction to manage internal and external network integration and service delivery.

Therefore, we address the question: what is the strategy of the telco to manage and benefit from telecom-IT convergence? The telco's strategy is to define a generic *business model* that can accommodate various external roleplayers. These players being other telcos or IT-using enterprises. By extending the generic business model the telco define specific value chains. Within these value chains the telco and the various external roleplayers define business objectives. Business objectives are satisfied by enabling telco infrastructure and external infrastructure to interwork. To enable this interworking the telco must *evolve* its network into a service centric network that uses the SDP as an infrastructure *integration mechanism* as suggested in *Figure 1.3*. Hence, the SDP must manage telecom-IT convergence.

The SDP enables telco infrastructure integration with IT-based infrastructures by adhering to the convergence process, that is, it defines points of integration across the telco infrastructure. These points of integration provide external roleplayers with access to a collection of software-based *services*. These services *abstract* telco infrastructure complexities and can be *used* for application development and content delivery.

The SDP has no agreed definition nor is it standardised. This poses a problem since external IT-using enterprises that access multiple SDP implementations do not have consistent access to a set of standard-based services. As a result, points of integration on the telco infrastructure are not standardised. Diverse SDP implementations also hinder *interoperability* between each other, since their points of integration are not standardised. This research proposes an approach to standardisation by defining *reference points* as SDP points of integration. Reference points prescribe a formal specification that is adhered to by their implementations. Hence, multiple SDP implementations that conform to their reference point specifications expose standard-based services and are interoperable.

To fully enable network integration, a collection of reference points are required for the telco infrastructure. These reference points formalise integration relationships between telco and external IT-based roleplayers. The roleplayers external infrastructure is abstracted as independent service entities that use SDP services. These entities may relate to converged business model entities.

An example set of reference points, structured within a SDP reference model, are shown in *Figure 2.6*. In the figure, service entities using the SDP via reference points are content providers and application providers. The SDP also uses reference points to abstract network capabilities and resources, such as transport networks, OSS/BSS and customer terminals. Based on various forms of convergence, we classify reference points as either *horizontal* or *vertical*. In this research we focus on reference points offered by the SDP. We name

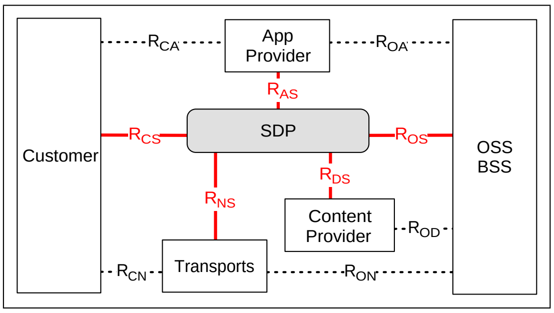


Figure 2.6: Proposal for a SDP Reference Model with Reference Points

these reference points *service oriented reference points*. Two horizontal service oriented reference points are:

- R_{CS} - customer to SDP.
- R_{OS} - OSS/BSS to SDP.

Three vertical service oriented reference points are:

- R_{AS} - application provider to SDP.
- R_{NS} - transport networks to SDP.
- R_{DS} - content provider to SDP.

By decomposing service oriented reference points we produce details on the SDP that implement telco and IT infrastructure integration.

The remaining reference points define integration relationships between roleplayers, their infrastructure and telco infrastructure. The R_{CA} and R_{CN} reference points define the customer relationship with the application provider and transport respectively. The R_{OA}, R_{OD} and R_{ON} reference points define the OSS/BSS relationships with the application provider, content provider and transport networks respectively.

2.4 Summary

In this chapter we discussed convergence between telco and IT infrastructures. We defined convergence as a process that separates infrastructure according to its functions performed, so as to identify points of integration. These points of integration enable infrastructure to converge. Using the convergence process, we have separated network infrastructures into

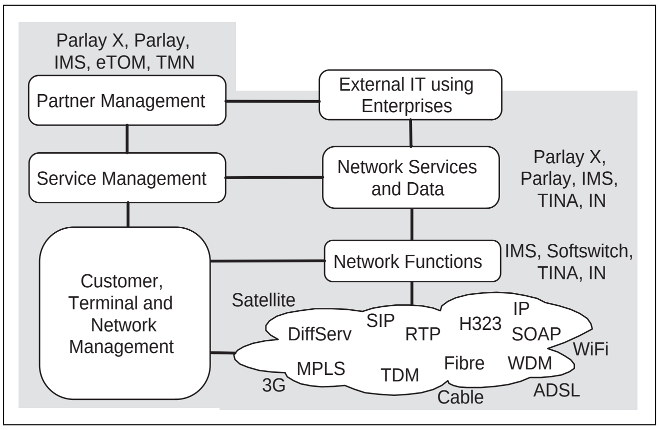


Figure 2.7: Complexity of Convergence

abstract layers and domains. Based on these abstractions, we have defined vertical and horizontal convergence occurring within and across network domains respectively. We have also described the implications of convergence on telco infrastructure. These include creation of value chains and integrating signalling, transport networks, service platforms, application environments and management systems. Associated with these implications we have described technologies and their architectures that contribute to telecom-IT convergence. These technologies contribute to the complexity of telecom-IT convergence. *Figure 2.7* illustrates these technologies against the various areas of the telco network. Due to convergence complexities we have defined a strategy for the telco, that is, development of the SDP as an integration mechanism between telco and IT-based infrastructures. We defined the SDP to provide points of integration that are formalised as a set of reference points. These reference points are structured within a reference model. The reference model entities relate to converged networks business model entities shown in *Figure 2.2*. Reference points are implementable as a set of abstract services. These services enable IT integration with telco network capabilities and resources. Hence, the SDP abstracts lower network related technologies and structures higher service related areas, that are shown in *Figure 2.7*. To fill out the detail we examine several existing standards in the following chapters:

Chapter 4: IN and TINA.

Chapter 5: Parlay and Parlay X.

Chapter 6: TMN, TOM and eTOM.

Chapter 7: SOA.

Chapter 8: IMS.

Chapter 3

Managing the Complexity of the Service Delivery Platform

The SDP has been located in its working environment in *Figure 2.6* but is, at this stage, a concept that is not yet standardised. As a result, it has many interpretations that produce varied definitions, designs and implementations. Each interpretation focuses on specific requirements, such as service creation using web services [7], content management and consolidating billing platforms. These interpretations also use diverse technologies to implement the SDP.

A popular interpretation, from [8], defines a limited set of requirements that prescribe the SDP as:

- managing service creation, provisioning, execution and billing;
- supporting the delivery of services in a network and device-independent manner;
- providing a single standardised point for application developers to find and use diverse services and content; and
- providing external developers, such as IT-using enterprises, with open and secure access to telco network capabilities.

Building on these requirements, we define the SDP as *"a distributed IT-based system that abstracts telco network capabilities into generic services that are accessible across telco, enterprise and Internet networks and promotes the development, delivery and management of various customer services"*. In this work we use this definition for the SDP.

Accompanying the above requirements is an SDP architecture that is defined in [8] and illustrated in *Figure 1.3*. In this work, we provide additional detail on the architecture

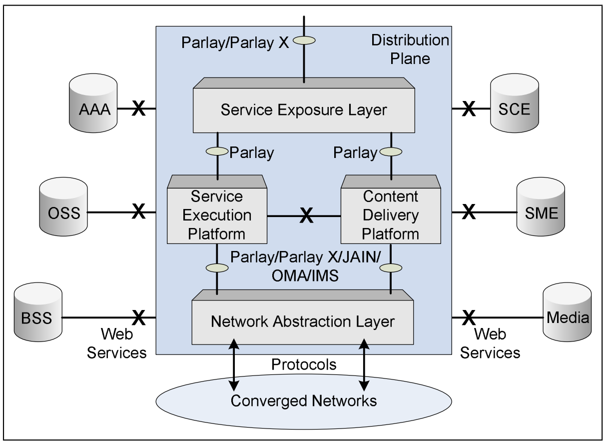


Figure 3.1: Expanded SDP Interpretation

to include interfaces, implementation technology mappings and a distribution plane. The added interfaces illustrate the communication between layers, platforms and telco resources. The technology mappings reuse existing telecom standards to show implementation choices for each interface, layer and platform. The distribution plane abstracts the distribution of the SDP and integration with other telco platforms. The expanded architecture is shown in *Figure 3.1*.

In the figure we illustrate existing standard-based technologies and APIs implementing platform and layer communication. For example, both Parlay and Parlay X [20] implement the service exposure layer API. However, Parlay also implements the network abstraction layer, execution platform and content delivery platform APIs. In addition, the network abstraction layer API is implemented using the standards-based technologies such as:

- Parlay X,
- Java API for Integrated Networks (JAIN) [49],
- Open Mobile Alliance (OMA) Service Environment [50, 51] and
- Internet Protocol Multimedia Subsystem (IMS) [24].

In *Figure 3.1* no suitable standardised technologies are defined to implement OSS/BSS, SCE and SME APIs. Also, communication across the service execution and content delivery platforms use proprietary APIs. We identify the need for standard-based APIs as X's in the figure.

3.1 Limitations

The SDP architecture in *Figure 1.3* is derived from various *products*, such as [52, 53, 54, 55, 56, 57], and is influenced by their specific requirements and architectures. In addition, the products use mixtures of telecom, IT and broadcasting technologies to implement the SDP. These technologies are both proprietary and standards-based. Hence, the SDP architecture in *Figure 1.3* is a generalisation of common functions offered by various products.

The SDP in *Figure 1.3* is *incomplete*, since layers and platforms require further decomposition to uncover additional functionality. These functions may include additional layers that manage information contained in various data-stores. Also, platforms may be decomposed to show additional components that manage distribution, security and replication of data.

The architecture does not provide an appropriate device (customer) domain. As a result, communication between device, SDP and converged networks are not managed. Also, content provider platforms or layers are not shown; rather, their infrastructure is directly integrated into the SDP. Hence, communication between content providers and SDP are also not managed or shown.

Additional abstractions are defined by decomposing the SDP architecture in *Figure 1.3*. These abstractions may provide details on business entities or implementation specifics. Also, subsets of abstractions may be structured into specific architectures, such as a business model or functional schematic. Hence, the architecture is limited since it provides a single abstract view on the SDP.

Figure 3.1 shows the SDP architecture in *Figure 1.3* being implemented using proprietary and standards-based technologies and APIs. However, different APIs implement a single platform or layer. In addition, proprietary web-based APIs are used to expose OSS/BSS functionality to layers and platforms. By using mixtures of standards-based and proprietary technologies *inconsistency* among SDP implementations occur. Thus, SDP implementations are not *portable* across converged networks and services are not portable across SDPs.

The SDP is a distributed platform that operates within a distributed environment. This environment is supported by a distribution plane. The plane supports communication between diverse implementations of layers, platforms, services and APIs. However, [8] does not provide *detail* on the distribution of the SDP, the distribution plane and its technology choices.

3.2 Approach

To overcome current SDP limitations we require a common base of concepts. These concepts contribute towards a generic SDP definition, set of requirements and an architecture. These concepts are void of any technology biases and promote the standardisation of the SDP. To defined and encapsulate these concepts we develop the *SDP framework*.

Traditionally, a framework provides a generic, extendable and supporting structure that describes a set of concepts, principles, rules, methods and technologies used to complete a specific task or process [58, 59]. For the SDP, we define its framework as providing a technology neutral foundation of reusable concepts and structures on which SDPs are designed, developed and implemented. To provide these capabilities the SDP framework prescribes various requirements.

3.2.1 Requirements

Like the convergence process, the SDP framework *separates* complexities from current SDP interpretations and simplifies then into varying *abstractions*. These abstractions are technology neutral and defined in an implementation and distribution independent manner. Hence, the SDP framework does not specify technologies, but rather *service-oriented* abstractions.

The framework contains various levels of abstractions that are integrated together to form a SDP *architecture*. Integrating abstractions is done in a generic manner. This maintains implementation and distribution independence. Hence, the SDP architecture is technology neutral and structures the various service-oriented concepts.

By applying the SDP framework, an architecture of abstractions is created. Some abstractions are generic, while others are specific to the SDP. Generic abstractions may be decomposed, extended or reduced to uncover particular SDP details. For instance, data processing abstractions may be grouped, decomposed and structured within a data-oriented SDP architecture. Therefore, the SDP framework and its resulting architecture are highly *extensible*, due to its technology neutrality.

The SDP framework represents a tool for managing the complexity of SDP development and therefore telecom-IT convergence. Complexity is abstracted into a generic architecture based on the SDP framework. This architecture provides varying levels of detail on the SDP that are technology independent. Therefore, the framework abstracts technological concerns without obscuring important SDP concepts.

To develop the SDP framework we treat the SDP as a complex system. As a result, we reuse and extend complex system concepts to define the SDP framework.

3.3 Complex Systems

Like the telco network, the SDP is a complex system. Complex systems are defined as a complicated interworking of components that share a common purpose [60]. The purpose of a system is based on the problem the system solves. Thus, the SDP contains a variety of heterogenous components that aim to provide various telco, IT and content-based services to fixed and mobile customers in a reliable, secure and quality assured manner.

Some system components are not monolithic entities, rather they are complex systems themselves. These components are known as "systems of systems" [61] or "subsystems" [60, 62]. Similarly, SDP components represent complex subsystems, since they simplify and integrate multiple network components to manage telecom-IT convergence. Hence, SDP subsystems provide *service-oriented* functionality.

Complex systems and their components have various properties. These properties are shared by the SDP and its components. For instance, components interact to fulfil the system's purpose. Interactions are formalised using relationships [62]. Examples of relationships include producer-consumer, client-server and peer-to-peer. These relationships define rules, policies and attributes to limit or enhance the functionality of interacting components. Hence, relationships structure and formalise interactions between components. Therefore, formalised relationships between SDP components aids its *standardisation*.

Like other complex telco systems, the SDP is an open [62, 63] system in the sense that it interacts with its environment of converged telecom-IT networks through standard interfaces. SDP and environment interactions include the exchange of data, protocols, API invocations and execution of business processes. By being open the SDP is also a dynamic [62] system. Dynamic systems contain components that interact with their environment. These interactions may cause components to change in many ways. For example, a call control component's state may change due to interactions with customers in the environment.

Systems engineering concepts are used to define a complex system's components, such that they satisfy system properties. These concepts are applicable to the SDP and the development of its framework.

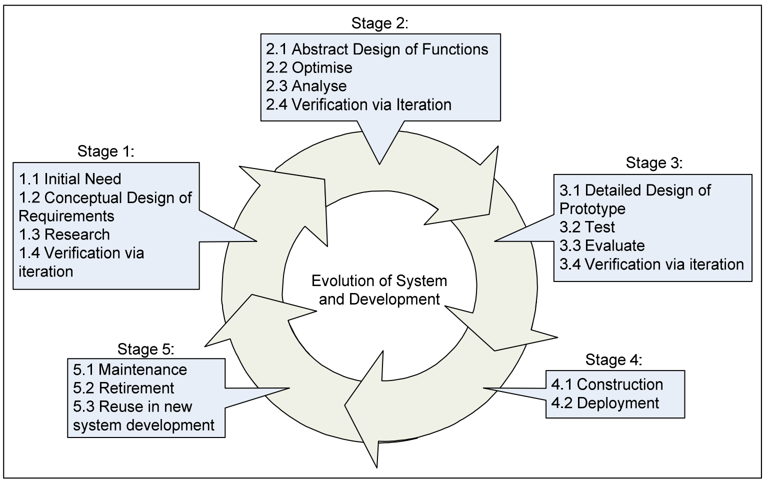


Figure 3.2: Simplified Systems Life Cycle

3.3.1 Managing Complexity

An approach used to manage system complexity is abstraction. Abstraction is defined as *"a way to do decomposition productively by changing the level of detail to be considered."* [64]. Thus, abstraction enables one to hide or focus on certain details so as to simplify a system and its components. Using abstraction provides a *perspective* on the system by hiding or focusing on specific system details.

Examples of abstractions include functional decomposition, reference points, layers, domains, planes, services and interfaces. These abstractions are used to describe generic concepts found in various telco network standards. For instance, [26, 65, 66] use abstractions to structure generic concepts contained in various telco transport networks, switching and service platform standards.

Various fields of research use abstraction to develop large complex systems. For example, [63] defines a systems thinking approach that uses abstraction to decompose a system into various parts, with each part being further decomposed. In modern systems engineering approaches abstraction is used within a process called the system life cycle [60]. The life cycle defines various stages that enable the systems and its parts to be defined, developed and deployed.

A simplified representation of a system's life cycle, derived from [62], is shown in *Figure 3.2*. In the figure, initial stages provide functional abstractions by decomposing the system according to the functions performed. Latter stages of the life cycle provide an implementation by specifying technology details a maintenance strategy. Each stage provides a specific *perspective* on the system and its development. Within each stage abstraction is used to hide or express details applicable to the specific perspective. For example, initial stages focus more on functional characteristics of the system, while latter stages focus on technological realisation of the system.

Tools such as methodologies, methods and models are used to abstract system details [67]. These tools complement the systems life cycle and its stages. Methodologies represent a grouping of methods used to solve a problem or develop a system. Methods define a logical process that is applied on a system to decompose it into specific abstractions. By applying a method a model is produced. A model illustrates a specific level of abstraction on the complex system by structuring its abstractions.

The system life cycle and its stages are implementable using various methodologies. However, in this research we use the system life cycle as an overall methodology to develop complex systems, such as the SDP. In addition, we incorporate methods as stages of the system life cycle to define system abstractions. These abstractions are structured into models.

3.4 Modeling Complexity

In the system life cycle, [60] defines models as producing *architectures*. Architectures represent the abstract linking, organisation, structuring and decomposition of the system and its parts [60, 68, 69]. Architectures are the most crucial element to managing telco system complexity [65]. In addition, architectures may be used to structure a specific system perspective, that is, arrange its abstractions.

Since the SDP is an IT-based system [8], its architecture is founded on software-based concepts and produces varying levels of software-based abstractions. Due to the IT-based nature of the SDP, we focus on software-based architectures for the SDP framework.

Software-based architecture is defined as the "structure or structures of the system, which comprises software elements, the externally visible properties of those elements, and their relationships" [68]. Within the telco domain, [70] defines software-based architectures as being reusable, extendable and separated from technology concerns.

Various types of software-based architectures exist that satisfy the above properties and are applicable within the systems life cycle. For example, [71] defines four generic categories that software architectures fit into with specially coined names, that is:

1. Decision: architecture is used as a strategy to manage a system under development.

This architecture is defined during the initial stages of the system life cycle and represents a highly abstract representation of the system.

2. Language[1]: abstract architecture representing the system's generic structure. This architecture is defined within the initial stages of the system life cycle.

3. Literature: system architecture that is reusable in other system developments. This architecture is defined by previous system life cycles, but provides reusable abstractions that are applicable to other system architectures.

4. Blueprint: represents an implementation-specific architecture that describes a system's technological realisation. This architecture is defined during the latter stages of the system life cycle.

In this typology, the SDP framework represents a technology neutral architecture that is both language and literature based. As a language, the SDP framework provides a common technology-neutral architecture that is used throughout SDP development. In addition, the architecture provides a generic overview of the SDP's structure. As literature, the SDP provides a technology neutral foundation of concepts and abstractions that are reusable in other service-oriented systems architecture. Also, the architecture is extendable to suit other system architectures. In both cases the SDP framework satisfies the generic architecture properties.

Software-based architectures are defined by applying various viewpoints (or perspectives) on the complex software-based system. Similar to abstraction, viewpoints express the "separation of concerns" [72] of a particular software-based system. Thus, a viewpoint focuses on particular system concerns while hiding others, so as to define abstractions and structure associated architectures. Viewpoints are similar to methods since their application produces a level of system abstraction contained within a model. Hence, viewpoints may be used as stages of a system life cycle.

In various domains viewpoints are defined to aid system development. These viewpoints are contained within a standardised methodology that is applied to system development. Examples of viewpoints and their methodologies include the Reference Model for Open Distributed Processing (RM-ODP) [72], Model Driven Architecture (MDA) [73] and the NGOSS [48] initiative. These viewpoints may be used as stages of a system's life cycle.

Viewpoints are chosen before system development starts. According to [74], viewpoints are chosen based on the following system properties:

[1]Language is not used in the sense of a programming lanuage

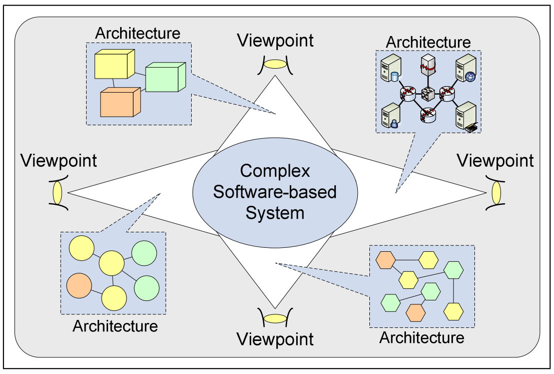


Figure 3.3: Abstracting Complex Software-based Systems

- Environment: includes the relationships between the larger system containing the system under development.
- Business Drivers: the business objectives the system aims to satisfy within its environment.
- Organisation: the inherent functions and properties of the system prescribed by the controlling organisation.
- Technology: the requirements for the system implementation.

We integrate methodology, method, viewpoint, model and architecture concepts in *Figure 3.3*. The figure shows a methodology containing various methods, called viewpoints. Applying viewpoints to the software-based system produces abstractions that are contained within a model, that is, software-based architecture.

By applying these software architecture concepts for complex systems, we present the approach used to define the SDP framework. The approach defines various viewpoints on the SDP. Each viewpoint considers a specific standard-based technology that contributes abstractions towards the SDP framework. Within each viewpoint we use the abstractions to structure a technology neutral SDP architecture. These architectures contribute towards solving the problems identified in *Chapter 1 Section 1.3*.

3.5 SDP Framework Development

The SDP framework provides varied levels of abstractions that are structured into an architecture. Based on properties of telecom architectures, the SDP framework must reuse other standards-based telecom architectures to reduce SDP complexity. The framework must also be generic, reusable and extendable. The framework must be implementable using any standardised telecom and IT technologies, hence, its design must remain technology independent. In addition, the architecture must promote SDP standardisation.

For the SDP framework to define generic SDP concepts and the associated architecture we require viewpoints. To define the viewpoints we determine its environmental, business, organisational and technological properties:

- Environment: the SDP operates within a converged ICT market and operates across telco, IT and Internet networks. This market is regulated and provides an open, competitive and service-rich environment. As a result, the SDP may partner with various other enterprises, such as IT-using enterprises and content providers.
- Business Drivers: the SDP provides services to customers. These services deliver both single-media and multimedia services. In addition, services are used by various 3rd parties, such as application developers, content providers and brokers. Also, the SDP simplifies service related business and operational process for the telco and its partners.
- Telecom-based Organisation: the SDP is defined by the telco to leverage existing systems and enhances them with IT-based functionality. In addition, the SDP is telco grade. The SDP is able to expose telco network capabilities to 3rd parties using IT-based technologies. As result, the SDP supports and delivers a varied range of services to telco customers and partners.
- Technology: the SDP is standards-based. As a result, multiple SDP implementations are operable across converged networks. Also, services are easily portable across SDP implementations. Hence, SDP specifications are technology neutral and implementable using a variety of standardised technologies.

From the above properties we define the following technology neutral viewpoints, that are helpful in simplifying specific SDP complexities, extracting abstractions and structuring various architectures. These viewpoints are also influenced by various existing standard-based telecom and IT architectures. These architectures provide a wealth of concepts and abstractions that can be extracted and extended to the development of the SDP framework.

- Legacy Perspective: includes legacy telco architectures that contribute to the SDP concept. This viewpoint relates to the SDP organisation property.
- Services Perspective: reuses current service platform architectures to uncover SDP concepts. This viewpoint relates to the SDP business property.

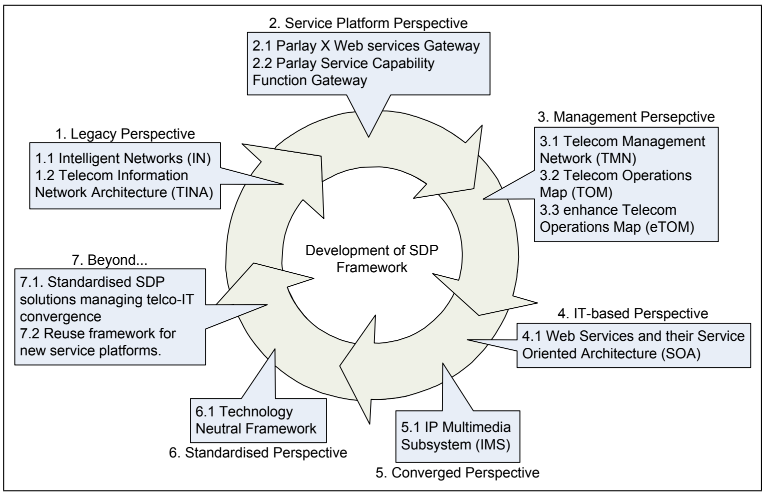


Figure 3.4: Approach to Develop SDP Framework

- Management Perspective: extracts generic concepts from management architectures and aims to incorporate them into the SDP framework. This viewpoint relates to the SDP environment and organisation properties.
- External IT-based Perspective: integrates IT-based architecture concepts into the SDP framework. This viewpoint relates to the SDP business and organisation properties.
- Converged Perspective: uncovers generic service-oriented concepts from a current telco network architecture that aims to manage telecom and Internet convergence. This viewpoint relates to the SDP environment property.

All viewpoints review standards-based technologies. To adhere to the SDP technology property, we extract technology neutral concepts and architectures from each technology and integrate them into the SDP framework.

Besides the SDP framework, the perspectives represent a time-line of architectures and technologies that contribute to the current SDP concept. As a result, we group the perspectives into a SDP development methodology shown in *Figure 3.4*.

In the figure the sixth perspective represents the integration of the generic concepts and abstractions obtained from previous perspectives into the SDP framework. The remaining perspective illustrates the reuse of the SDP framework for the development of SDP implementations or other service-oriented systems.

3.6 Summary

In this chapter we have discussed the current SDP interpretation. We have highlighted this interpretation's definition, requirements and architecture. Also, we have identified its limitations such as technology dependence and lack of standardisation. We have defined the SDP framework as the solution to these limitations. The framework aims to simplify the complexity of developing a SDP. We have shown the SDP is a complex system. Also, we have reused system engineering concepts to define the approach used to develop the SDP framework. We have discussed concepts such as abstraction, models, methods and methodologies. In addition, we provided a systems life cycle approach to encapsulate these concepts. We have also shown architecture as an important tool that contributes to managing SDP complexities and developing the SDP framework. We have discussed viewpoints as a means to create architectures. We have associated models to architectures, viewpoints to methods and methodology to the system life cycle. Based on SDP properties we have defined a set of viewpoints, that is, legacy, services, management, IT-based and converged perspectives. These viewpoints form the basis for chapters 4 to 8. The viewpoints extract generic and technology neutral concepts from various telecom and IT standards. Concepts are structured into appropriate architectures. Integrating concepts and architectures represents the SDP framework. This framework is the essential step toward answering the problem and subproblems stated in *Chapter 1 Section 1.3*.

Chapter 4

Perspectives on the SDP from Legacy Standards: IN and TINA

Traditionally, telco services are implemented in software that is tightly *coupled* with switching hardware. This results in both standardised and proprietary software and hardware being developed by different vendors. As a result, customer services are limited and developed slowly. Also, service and network administration is a complex task that is managed by integrating additional vendor software and hardware into the network. Thus, the traditional telco is faced with highly integrated vendor-specific solutions that *limit* service development, delivery and management.

To overcome the above limitations, the telco network operator requires a simpler approach for service creation, management and delivery. Hence, the need for service and network *separation*, supporting flexible service and network integration. This results in service software being separated, to a certain degree, from switching hardware and placed on a dedicated service network.

The Intelligent Network (IN) was defined to aid in the development of a separate but integrated service network. With further innovation and use of middleware-based technologies the Telecommunication Information Network Architecture (TINA) was defined, but not deployed by telcos. Both IN and TINA share similarities with the SDP, such as their service requirements. In the following sections we discuss the IN, TINA and their contributions to the SDP and its framework with the objective of uncovering abstractions that contribute to a technology neutral SDP architecture.

4.1 Intelligent Network

The IN aims at enabling the "provision of services independent of the service/network implementation in a multi-vendor environment" [2]. IN services are predominantly *voice-based* in switched circuit networks. Also, the IN shares telco properties such as being reliable, secure, managed and providing the necessary quality of service.

The IN strives for vendor *independence* by removing service implementations from switching hardware and placing them on a separate overlay service network. The service network operates over circuit-mode transport networks and is exposed to external parties, such as service developers. Though contemplated in the IN standards, network exposure to external parties is not fully defined in the IN.

The IN is implemented as a *distributed* service platform that promotes customer service development independent of the underlying telco infrastructure implementations. This independence is gained by abstracting telco infrastructure capabilities into reusable software-based *service* logic. The service logic is integrated to create customer service implementations. Also, the IN supports the delivery of these services to customers on telco transport networks.

4.1.1 Requirements

The IN aims at enabling the separation of generic service logic from vendor-specific hardware, with the intention of integrating both in an effective and technology independent manner. As a result, the IN provides the *glue* between services and network resources, with little or no emphasis placed on open external access. These IN properties are illustrated in *Figure 4.1*.

The figure depicts the IN as a *point of integration* between services and network resources. Thus, the IN is used by both service developers and the network operator. As a result, the IN must satisfy both service requirements and network requirements. As defined in [2], both service and network requirements are based on:

- service creation: service providers create service logic that makes use of network capabilities and resources, such as establish connections and play announcements. The network operator provides service creators with programmable access to these network capabilities and resources.
- service management: service providers require network resources to make provision

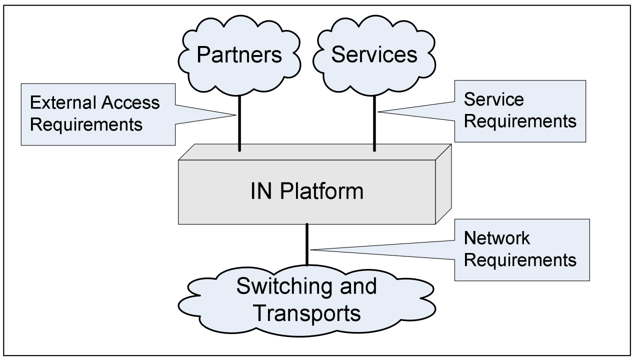


Figure 4.1: IN Requirements Classification

for new services and to manage new services, such as user profiles and billing. The network operator provides these network capabilities and may offer external access to these capabilities.

- service processing: service providers require the network to correctly process service logic and network resources, so as to deliver services to customers. The network operator supports this processing, by ensuring network resources are used appropriately and provide necessary service functionality.
- network management: the network operator requires the complete administrations and management of services, IN platform and network resources.
- network interworking: the network operator requires service usage to be independent of the customers access and network.

To satisfy both service and network requirements (but not management), a reference model is defined, that is, the IN *conceptual model*. The IN definition, concepts and architecture is encapsulated in the conceptual model.

4.1.2 Architecture

The conceptual model represents a reference architecture for the IN. Also, it provides a foundation for the definition of IN standards. As a result, IN-compliant architectures are derived from the conceptual model and adhere to its principles. Hence, the IN conceptual model represents a meta-architecture or *framework* that is used in the development of IN-based systems. Hence, the IN conceptual model is the main contribution of the IN to the definition of the SDP and its framework.

The IN conceptual model provides different *perspectives* on the IN. These perspectives are applied to the IN in a *top-down* approach. Each perspective is represented as a plane that

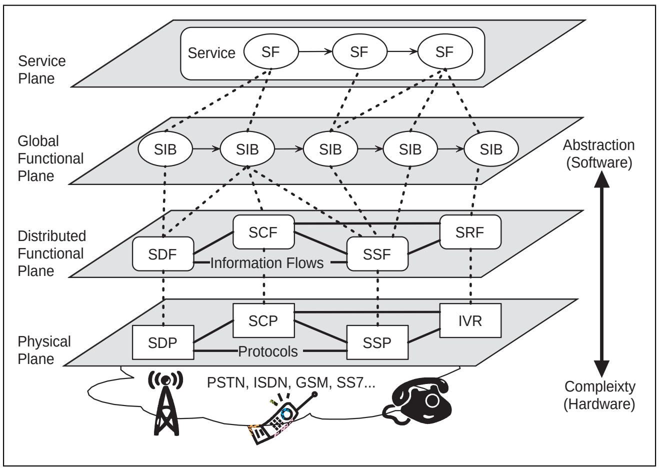


Figure 4.2: IN Conceptual Model

contributes to the structure of the IN. The IN conceptual model is illustrated in *Figure 4.2*. The figure depicts the various planes:

- Service Plane: defines capabilities offered by the IN platform as abstract service features that integrate to form customer services. This plane does not define service feature or customer service implementations.
- Global Functional Plane: defines reusable service logic called *Service Independent Building Blocks (SIBs)*. An integrated chain of SIBs implement a service feature. Hence, a collection of integrated SIBS implement a customer service.
- Distributed Functional Plane: defines distributed telco *functions* used to implement SIBS. Functions are contained within functional entities. Functional entities implement SIBS by communicating via information flows to execute specific functions.
- Physical Plane: defines a strict mapping of functional entities to specific physical network elements. The plane also provides distribution and implementation details on the functional entities and their information flows. For example, INAP [75] implements the functional entity information flows.

4.1.3 Reusable Concepts

The IN increases the *intelligence*, rather than switching levels in the telco infrastructure [76]. The global functional and distributed functional planes are examples of this intelligence. The added intelligence abstracts the complexity of infrastructure technologies and distribution that are represented as physical elements in the physical plane.

The collection of planes represents the increase in *separation* between telco infrastructure, its capabilities and customer services. This separation enables customer service creation to be independent of the underlying infrastructure technologies. Also, this separation enables customer services to use infrastructure capabilities.

One of the IN philosophies is the *standardisation* of SIBs [77]. The standardisation promotes additional separation and independence between telco infrastructure and customer service definitions. For instance, changes to infrastructure do not effect SIB definitions and therefore do not effect service feature and customer service definitions. As an example, the charging SIB hides the network-specific mechanisms for writing billing tickets. However, telco infrastructure must provide the needed functions to implement SIBs. Therefore, conformance to SIB standards is achieved by providing necessary functions to implement SIB definitions.

Each plane of the conceptual model provides a different *perspective* on the IN. The service and global functional planes provide *service-oriented* views on the IN [78]. In these views technology and distribution of the telco infrastructure is hidden. The service plane enables service developers to verify the needed capabilities (service features) for a customer service. In the global functional plane developers view the integration of SIBs as the implementation of service features to create customer services.

The distributed functional and physical planes provide a *function-oriented* view on the IN. In this view, only functional entities and their functions, as offered by physical elements, are perceived. The functional entities and their functions represent capabilities offered by physical elements. This view also provides details on the technology and distribution of functional entities and physical elements.

Though customers are connected to the telco via its transport networks, customers perceive their *interactions* are with their services. The distributed functional and physical planes support this communication between customer and service.

An integrated *managed environment* is proposed for the IN. This managed environment is based on Telecommunications Management Network (TMN) [21] concepts. The general TMN architecture provides management layers that are incorporated into the IN planes. However, in practice vendors use non-TMN based solutions to manage the IN. The TMN and its contribution to the SDP is discussed in *Chapter 6*.

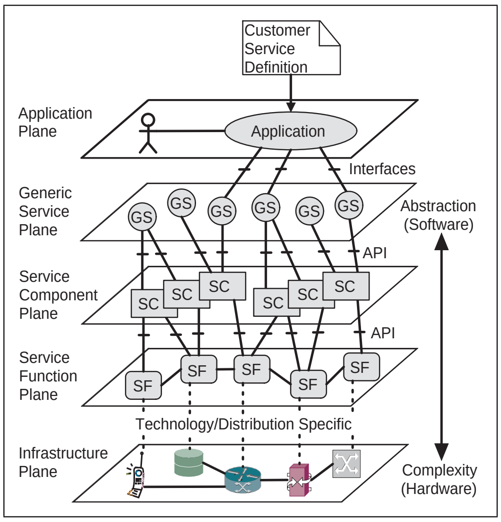


Figure 4.3: SDP and its Environment represented as a Conceptual Model

4.1.4 Contribution to SDP from IN

We reuse some of the IN conceptual model concepts, with variations, to define a SDP architecture. The variations are influenced by SDP requirements and current software paradigms, such as APIs. The SDP architecture is called the SDP conceptual model and is shown in *Figure 4.3*. The architecture structures the SDP within an environment. The SDP interacts with this environment using its service APIs.

The SDP conceptual model represents a viewpoint on the SDP that illustrates various technology neutral abstractions. These abstractions include the use top-down layered planes that provide perspectives on the SDP. Contained in these planes are numerous service building block and network function abstractions, that separate service intelligence from network specifics. We extend these service and functional abstractions to allow easier access to their capabilities. The SDP conceptual model is illustrated in *Figure 4.3*.

In *Figure 4.3*, customer services are defined by external application developers. The service definition documents the interactions between customer and service, as well as the capabilities required from the telco to implement the customer service. Once completed, the customer service definition is implemented as an application in the *application plane*. Thus, unlike the IN conceptual model's service plane the application plane contains the actual customer service implementation.

Applications use telco network capabilities by orchestrating *generic services* (GS). Generic services provide simplified access to telco network capabilities via their implementation independent interfaces. Generic services vary according to the telco network capabilities they simplify. For example, services may provide telecom, content or management-oriented functions. All generic services are grouped into the *generic service plane*. This plane supports the generic services and their interfaces that are exposed and used by applications.

Generic services abstract access to complex *service components* (SC) that invoke telco infrastructure resources and capabilities. Service component functionality is offered via an API. Thus, service components represent a common point of integration used to access telco network capabilities. Service components are housed in the *service component plane*. Thus, generic services, service components and their APIs provide greater levels of separation and network abstraction than the IN conceptual model SIBs.

Underlying telco infrastructure contains distributed, standardised and proprietary systems that are implemented and connected using various technologies. These systems provide the solid foundation of reusable resources and capabilities. To enable technology and distribution-independent access to these resources and capabilities, their functions are abstracted into services, named *service functions* (SF). Service functions offer their functionality to service components via an API. Also, service functions communicate amongst themselves to satisfy service component requests. Hence, the result of service function communication is the execution of functions on telco infrastructure. Service functions and their communication is limited to the *service function plane*. Therefore, unlike the IN conceptual model's functional entities the communicating service functions contribute to the implementation of the service components; rather than implement the service components.

In the SDP conceptual model, telco infrastructure is contained within physical elements and are connected on transport networks. These physical elements are managed in an *infrastructure plane* that enables their technology and distribution dependent functions to be accessed and used by service functions.

4.1.5 Evaluation of SDP Conceptual Model

By extending the IN conceptual model concepts to the SDP we have defined a conceptual viewpoint. This viewpoint defines various abstractions that are structured in the SDP conceptual model. The abstractions contribute to the definition of the SDP framework. We evaluate the SDP conceptual model against the generic concepts extracted from the IN conceptual model. In addition, the evaluation provides answers to the questions posed in *Chapter 1 Section 1.3*. We also present the comparison between IN and SDP conceptual models in *Table 4.1*.

The SDP conceptual model identifies external IT-using enterprises that use SDP services to create or enhance applications. These applications provide services to customers. No

<table>
<tr><th colspan="2">IN Conceptual Model</th><th colspan="2">SDP Conceptual Model</th></tr>
<tr><th>Plane</th><th>Description</th><th>Plane</th><th>Description</th></tr>
<tr><td>Service</td><td>customer services described by integrating service feature descriptions. No implementation.</td><td>Application</td><td>customer services are described and implemented by an enterprise. This is external to the telco and SDP.</td></tr>
<tr><td rowspan="2">Global Functional</td><td rowspan="2">building blocks (SIBs) define reusable service logic. Chains of SIBs implement service feature descriptions and therefore customer services.</td><td>Generic Service</td><td>application implementations use generic services via technology independent interfaces.</td></tr>
<tr><td>Service Component</td><td>service components offer an API to access service related telco network capabilities.</td></tr>
<tr><td>Distributed Functional</td><td>implement SIBs using distributed functional entities that execute functions via information flows.</td><td rowspan="2">Service Function</td><td rowspan="2">service functions offer an API to access telco infrastructure specific capabilities and resources. Also, service functions implement a communication mechanism to interact and satisfy service component request.</td></tr>
<tr><td rowspan="2">Physical</td><td rowspan="2">functional entities map to physical elements that implement their functions using specific technologies. Also, specific protocols implement functional entity information flows.</td></tr>
<tr><td>Infrastructure</td><td>connected and technology and distribution specific telco systems provide service functions with access to their specific functions.</td></tr>
</table>

Table 4.1: Comparison of Conceptual Models

business model is defined to accompany the SDP conceptual model, since the conceptual model limits business entities to external application developers.

We use the concept of planes as a design pattern to structure a SDP architecture, that is the SDP conceptual model. The SDP conceptual model uses planes to expose additional SDP details than the proposed SDP architecture, shown in *Figure 1.3*. Like the IN, the SDP conceptual model planes hide technology and distribution details of their applications or services. As a result, the SDP conceptual model inherits these technology and distribution neutral properties. In the conceptual model service related planes increase the levels of *intelligence* in the telco. Also, these service related planes are layered according to their level abstraction. For example, higher service planes abstract complex intelligence of lower planes, such that simpler access to telco network capabilities is provided to external developers.

The SDP conceptual model's hierarchy of planes and abundant services increase the *separation* and *independence* between telco infrastructure and application implementations. For example, applications using generic services are decoupled from the telco infrastructure. Thus, the telco infrastructure is adequately abstracted by the various levels of SDP services. All SDP services further support this infrastructure independence by providing access to their functions using implementation independent interfaces. Generic services expose their interfaces to application developers, while using less abstract service component interfaces. Service components use complex interfaces exposed by service functions to access telco infrastructure capabilities. Thus, the SDP conceptual model only exposes generic service interfaces to external IT-using enterprises, since they provide the most abstract access to telco infrastructure capabilities.

Similar to the IN, the SDP must implement its services and planes using *standards*. By implementing standards further independence of applications and telco infrastructure is gained. As a result, the telco conforms to these standards and ensures infrastructure can support standardised services. Also, developers benefit from standardised services and interfaces since they have a constant repository of reusable services for application development. Also, with standardised services and interfaces interactions between telco and external IT-using enterprises becomes standard-based. Candidate standards for generic services and their interfaces is Parlay X, while Parlay may implement service components and their interfaces. Also, the IMS [24] may implement a set of service functions, where SIP [37] implements horizontal service function communication. Additional standards may be used to implement remaining service functions and their communications. As a result, service planes may be implemented as distributed platforms that support the implementation of these standards.

Like the IN conceptual model, there are two perspectives created by the SDP conceptual model. The application, generic service and service component planes provide a *service-oriented* perspective on the SDP. For instance, customers view the telco as offering services (applications) and perceive they interact with these services directly. Also, developers view the generic service plane as a resource of generic services with interfaces to orchestrate into applications. The developer's view is limited to the generic service plane, while the telco is able to view all planes. For example, the telco views the service component plane as a resource of service components with APIs to orchestrate into generic services. The telco also views the service function plane as a resource of service functions with APIs to orchestrate into services components. In these service-oriented views service technologies, implementations and distribution is hidden by the planes and service interfaces.

The service function plane and infrastructure plane provides a *functional-oriented* view on the SDP. In this view the telco perceives the infrastructure plane as a resource of technology and distribution specific capabilities to be abstracted and integrated into services functions.

The SDP conceptual model incorporates the telco OSS/BSS, such that each plane and its services are managed within a *management environment*. Like the IN conceptual model, a managed environment administers applications accessing generic services, service interactions, all services and telco infrastructure. Each plane contributes functionality to the management environment. For instance, a service component plane implementation may provide management functionality to administer service components. Also, services in all service planes may abstract OSS/BSS functionality that is used in this management environment. Like the IN and TMN a separate management architecture may be defined, based on the SDP conceptual model. The management architecture may reuse the SDP conceptual model's technology neutral planes, services and interfaces. Thus, the management architecture may promote standard-based SCE and SME using the planes' services and interfaces.

4.2 Telecommunication Information Network Architecture

The IN reduced development time of new voice services, however, lack of complete standardisation especially for service creation and management results in IN systems being proprietary-based [79]. In addition, the IN does not include advances made in computing, information processing and multimedia. By incorporating these advances into the telco network new types of services, besides voice, are possible. Hence, to benefit from new technologies, overcome IN limitations and foster a complete and standardised telecommunication system, TINA [19] was defined in the 1990s. We examine TINA to develop a conceptual perspective on the SDP.

4.2.1 Requirements

As defined in [80], TINA provides a collection of concepts, principles and architectures that support the design, development and deployment of voice, information, multimedia and management services. In addition, TINA enables these services to operate over heterogenous transport networks and computing platforms through strong abstraction.

TINA focuses on *software-based* concepts and architectures. These architectures contain a wealth of reusable components that simplify various service-oriented complexities, such as service creation, testing, subscription, consumption and management. Also, TINA abstracts complexities introduced by the telco network, such as ensuring reliability, security and distribution, using software-based distributed computing technologies.

To define these concepts and architectures, TINA applies the RM-ODP [72] development process. By using RM-ODP, TINA is defined using various viewpoints. Applying each viewpoint produces various concepts and abstractions that are structured using a business

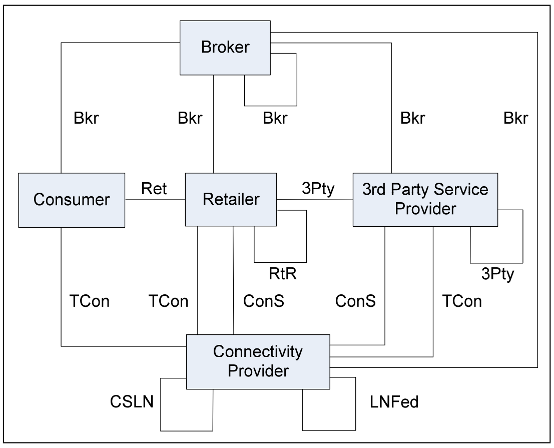


Figure 4.4: TINA Business Model

model and two architectures. The business model captures TINA's requirements, while the other architectures structure service and network abstractions. The TINA business model is illustrated in *Figure 4.4*.

The TINA business model defines communicating business entities that benefit from using a TINA-based service platform. These entities collectively use the TINA-based platform for various activities. These entities and their functions are:

- Consumer: locates, subscribes, consumes and pays for services offered by a retailer of service provider.
- Retailer: provides services to various consumers on behalf of itself or service providers.
- 3$^{\text{rd}}$ Party Service Provider: develops services that are offered by retailers to various consumers.
- Broker: is used by all entities to locate each other and their offered services. For instance, consumers locate retailers via brokers.
- Connectivity Provider: provides the underlying network resources and capabilities, such as reliable and secure transport networks.

These business entities also specify the business *domains* a TINA-based platform must operate across. For example, customer business entities operator within the customer domain.

In the business model, entities communicate to fulfill their activities. These modes of communication are formalised as *reference point* relationships. Reference points promote TINA standardisation by specifying the interactions between business entities. Thus, conformance to reference points ensures different TINA implementations are interoperable. TINA defines

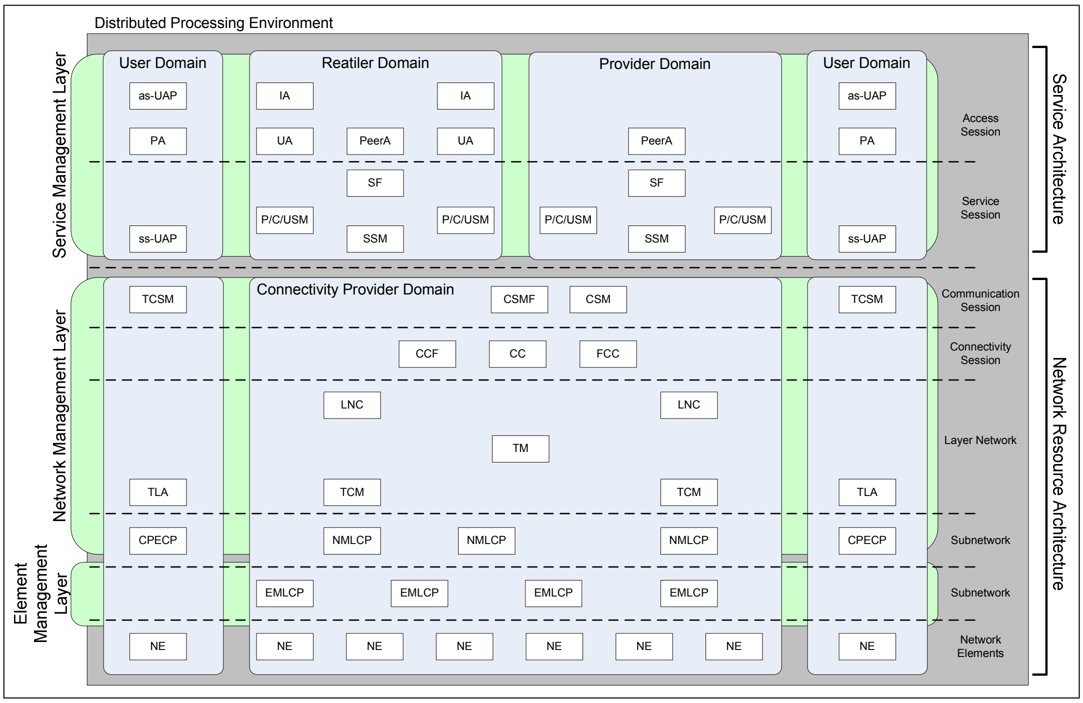


Figure 4.5: Simplified TINA Service and Network Resource Architectures (Interfaces not shown)

a service and network architecture by decomposing its business model, reference points and associated entities into components. This decomposition results from the RM-ODP computational viewpoint.

4.2.2 Architecture

TINA constitutes two standardised architectures: a service architecture defined in [44] and network resource architecture defined in [45]. The service architecture structures various component based service-oriented abstractions, while the network resource architecture structures component based network related abstractions. Both architectures are illustrated in *Figure 4.5*. The figure is derived from [44, 45, 66].

Both service and network architectures define software-based components containing rich data structures. TINA components also specify implementation independent *interfaces*. These interfaces hide component implementations and support component communication. These interfaces implement business model reference points and therefore promote TINA standardisation. Component interfaces and their communication are not shown in the figure due to their complexity.

In the service architecture, components are divided into access and service session categories. Access session components enable the customer to locate and activate services. Service session components provide the service to customers and support customer to service interactions.

Network resource architecture components support service architecture components by delivering connection services across diverse *transport networks*. The network resource architecture components abstract the complexity of communication, connectivity and network elements. Components abstracting customer equipment are also defined. Similar to service architecture components, the network resource architecture components operate across various domains.

Based on the TINA standards *Figure 4.5* divides both architectures into various *layers*, such as service session, communication and connectivity layers. We add the access session layer to the figure. In addition to these layers, TINA defines management layers. These are the service management, network management and element management layers. Within all management layers, both service and network components expose management functionality via their interfaces. These management functions are invoked by management applications to perform service, connection, fault and accounting management.

Layers are extended across functional *domains*. These domains correlate to the business entities identified in the TINA business model (shown in *Figure 4.4*). Functional domains imply various complexities, such as ensuring components operate over various technologies, across distributed domains and with diverse component implementations. Fortunately, standardised and implementation-neutral interfaces ensure interworking of components. Hence, interfaces enable interoperability of components across various domains. However, to provide the necessary technology and distribution independence, *middleware* is used. Middleware is defined as a set of software abstractions (services) used to hide various complexities associated with distribution systems, such as heterogenous hardware, diverse operating systems, numerous programming languages and diverse transport networks [81].

TINA supports service and network components by using a distributed computing mechanism. The mechanism supports technology, distribution and implementation independence of components and is named the Distributed Processing Environment (DPE). The DPE is a concept describing software-based middleware that contains generic and reusable services. These middleware services are used by components since they abstract underlying network complexities. The Object Management Group (OMG) [82] provides an open and standardised implementation for the TINA DPE, called the Common Object Request Broker Architecture (CORBA) [83].

4.2.3 Reusable Concepts

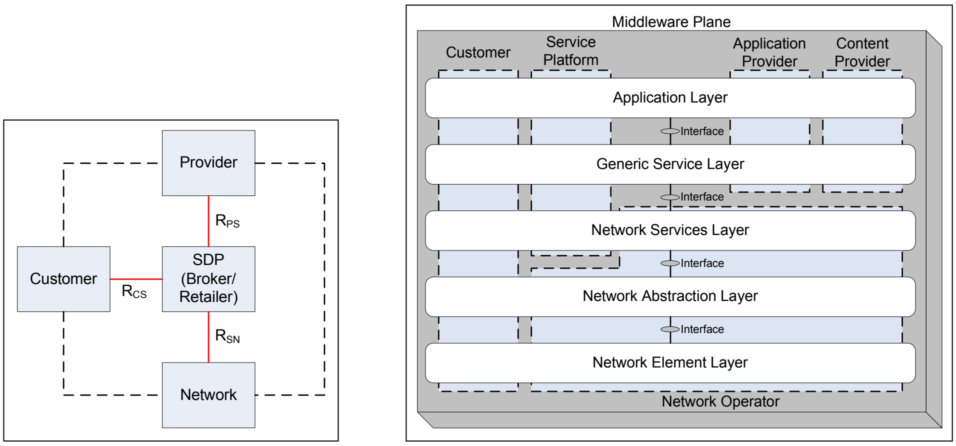
TINA provides various forms of *abstraction* that are reusable in the definition of the SDP and its framework. For instance, the business model represents a collection of requirements and entities that interact within a TINA environment. Also, the business model defines business domains that TINA operates across. Most importantly, the business model promotes standardisation by formalising business entity relationships as reference points.

Another form of abstraction used by TINA is that of *separation*. TINA applies the principle of separation by dividing itself into two perspectives, that are structured into service and network-oriented architectures [84]. The separation principle is also applied within these architectures, for example, the separation of service communication into access and service sessions. The principle is also applied to the network-oriented architecture since it separates communication and connectivity concepts from physical networks.

The TINA architectures are decomposable into layers [85] that group and manage components. The components represent abstract forms of reusable customer, service and network functionality. In addition, components separate access to their functionality by providing interfaces. Interfaces are technology neutral and hide component distribution and implementation details. Interfaces also expose component management functionality that is used by management systems and processes. TINA guarantees interoperability of component implementations since all interfaces are *standardised*.

The use of middleware is "one of the key assumptions made in the TINA architecture" [85]. Middleware abstracts technology concerns, such as implementation and distribution complexities. Also, middleware abstracts the interworking of TINA with legacy telco systems. As defined in [78], the DPE may abstract IN platforms. Hence, the DPE may be considered an extra plane that is added to the IN conceptual model. As a result, the DPE represents a service-oriented *middleware plane* that abstracts lower telco network capabilities into services.

Middleware services also provide management functionality to administer components. For example, some DPE implementations perform component life cycle management [45]. Management services and other middleware-based services are accessible and used by all components operating within the management plane.



(a) Generic SDP Business Model derived from TINA Business Model

(b) SDP and its Environment derived from TINA Architectures

Figure 4.6: Reusing TINA Concepts for the SDP

4.2.4 Contribution to the SDP from TINA

By reusing and extending the generic TINA concepts, independent of any technology bias, we derive a SDP business model and architecture. The business model and architecture are shown in *Figure 4.6*.

A SDP business model, shown in *Figure 4.6(a)*, is derived from the TINA business model shown in *Figure 4.4*. The business model clearly demarcates the SDP and identifies its roles. The SDP business model does not define a stand-alone broker or retailer, but condenses the TINA business model's broker and retailer roles into the SDP. By performing these two business roles, the SDP is a central business entity within the business model. The remaining TINA business model's entities are reused but vary in roles and responsibilities.

As a retailer, the SDP offers services to customers and various 3^{rd} party providers. 3^{rd} party providers use SDP services to create or enhance customer services. Customers use SDP services to access customer services. Hence, the SDP conforms to the classical TINA retailer business role. As a broker, the SDP offers broker-related services to customers to locate, subscribe, consumer and pay for other services. Also, providers use the SDP's broker-related services to locate other service-offering entities within the business model. In TINA, the broker role is distributed across various access session components, connection layer components and the DPE. Thus, unlike TINA the SDP centralises the broker role within its business model.

Numerous providers are catered for in the derived SDP business model. For example, providers may include application providers or content providers. Application providers use SDP services to create single-media or multimedia-based services that are delivered to customers. Content providers use SDP services to create applications that either supply content to support application providers or deliver content-based services to customers. In the SDP business model customers may be individuals or enterprises that provide services to individuals. Thus, the SDP business model providers and customers relate to the TINA business model's 3$^{\text{rd}}$ party service provider and consumer respectively.

In addition to customers and providers, the SDP abstracts the complexity of the underlying telco network. These include the telco network resources and capabilities. These network abstractions are implemented as various SDP services. Thus, unlike the TINA business model the SDP business model includes a network rich with capabilities; rather than just connectivity.

Similar to the TINA business model, the SDP business model defines reference point relationships between business entities to promote standardisation. In the business model, reference points to the SDP are considered that is:

- R_{PS}: provider (application or content) to SDP relationship.
- R_{CS}: customer to SDP relationship.
- R_{SN}: network to SDP relationship.

The generic SDP business model, its entities and reference points may be *decomposed* to cater for specific SDP requirements. With further decomposition of the generic business model we define a SDP within its environment shown in *Figure 4.6(b)*. The figure structures the SDP and its environment as an architecture that is derived from the TINA architectures shown in *Figure 4.5*. In the SDP architecture we use the concept of layers to separate SDP functionality. These layers are:

- application layer: houses applications developed by providers. Also, includes content and content processing applications managed by content providers. These applications provide similar functions as the components contained in the TINA access session and service session layers. Thus, the application layer is similar but not identical to the TINA 3$^{\text{rd}}$ party service provider.
- generic services layer: groups services that provide generic functionality such as subscription and session management. These services are used by provider applications. TINA does not define separate generic services for its service architecture, but similar service functionality is distributed across its access session and service session layer components. For instance, the generic services layer represents a more detailed splitting of the SSM logic from the lower network communication logic.

- network services layer: exposes telco network resource and capability functions. These services offer their functionality independently of technologies. These services provide similar abstractions as components found in the TINA communication session layer. For example, the network services layer represents the strict decoupling between higher layer services and lower CSM logic.
- network abstraction layer: provides a single point of integration between abstract network services and physical network resources and capabilities. These services abstract the complexity of underlying telco network systems. These services provide relatively simpler abstractions than components found in TINA network resource architecture's connectivity session, layer network and subnetwork layers.
- network elements layer: represents the physical network systems that are used to support and deliver services to customers across converged networks. This layer is similar to the TINA network element layer.

Lower SDP architecture layers expose their services to higher layers using technology neutral interfaces. As with TINA components, the SDP services hide their implementation and distribution behind their interfaces. These interfaces also abstract access to the service functionality. Service interfaces implement the SDP business model's reference points, so as to promote standardised communication between business entities. This is also seen in TINA with its component interfaces implementing its business model reference points.

Services and their interfaces, contained within the SDP architecture layers, are distributed across domains. Like TINA, these domains represent areas of *functional* division that relate to the business model entities. Thus, the SDP architecture layers are distributed across customer, SDP (service platform), network and provider (application and content) domains. These domains relate to the SDP business model entities shown in *Figure 4.6(a)*. These domains are decomposable into other specific domains, such as a network access, edge and core domains. In addition, an OSS/BSS domain may be extracted from the network domain. Abstracting the complexities associated with technologies, implementations and distribution found in the SDP architecture is the *middleware plane*.

The middleware plane hides technology, implementation and distribution concerns from all applications and SDP services. For instance, the middleware plane supports service communication across layers and domains. In addition, the plane hides implementation details of communicating services. Also, the plane abstracts transport networks and generic *functionality* provided by some of the underlying telco network systems. These network systems may include other service platforms, middleware and databases. The middleware plane also provides its own services that are used by other services and applications in the SDP architecture. Like all SDP services, the middleware plane also promotes technology independence by offering implementation independent interfaces to its own services. Therefore, the SDP middleware plane provides functionality similar to the TINA DPE.

4.2.5 Evaluation of SDP Business Model and Architecture

By extracting, extending and applying the generic TINA concepts to the SDP we have defined a conceptual viewpoint. This viewpoint defines various abstractions that are structured in the SDP business model and architecture. The abstractions contribute to the definition of the SDP framework. We evaluate the SDP business model and architecture against the generic concepts extracted from TINA. In addition, the evaluation provides answers to the questions posed in *Chapter 1 Section 1.3*. We also present a comparison between TINA and SDP concepts and architectures in *Table 4.2*.

The SDP business model represents a collection of SDP *requirements*. In addition, it illustrates the role players involved in the converged telecom-IT environment. The business model also promotes interoperability between SDP implementations by enforcing reference point relationships between the various role players. Reference points promote *standardisation* of the SDP and its interactions with customers, providers and telco network. Therefore, customers have universal access to their services, applications are portable across SDP implementations and SDP implementations are portable across telco networks.

The SDP architecture shown in *Figure 4.6(b)* represents the decomposition of the business model into functional *layers* and *domains*. Layers and domains represent design patterns used to structure the SDP architecture independently of technologies. Layers group hierarchies of SDP *services*, while domains distribute layers and their services across the converged telco-IT networks. Both layers and domains may be separated into more specific layers and domains. The SDP architecture identifies additional layers and introduces domain distribution to the proposed SDP architecture, shown in *Figure 1.3*.

Like TINA components, the numerous SDP services represent strong abstractions of telco network functionality. In addition, some services abstract customer-related functions. All SDP services communicate in a distribution and technology independent manner, by using their implementation independent *interfaces*. These interfaces support communication across layers and domains. Service interfaces also provide authorised access to a component's management functionality. This enables management systems, such as the telco OSS/BSS, to administer SDP services. However, service interfaces must be standardised since they implement business model reference points.

<table>
<tr><th></th><th>TINA</th><th colspan="2">SDP</th></tr>
<tr><td>Business Model</td><td colspan="3">Technology neutral representation. Encapsulates system's requirements. Defines business entities and domains that the system operates across. Unlike TINA, the SDP does not use an explicit broker or retailer.</td></tr>
<tr><td rowspan="2">Service Architecture</td><td>Access Session: Components abstract service access.</td><td rowspan="8">SDP Architecture</td><td rowspan="2">Application Layer and Generic Service Layer: Contains 3rd party provider applications and content that use generic services via their interfaces.</td></tr>
<tr><td>Service Session: Components abstract service usage.</td></tr>
<tr><td rowspan="6">Network Resource Architecture</td><td>Communication Session: Components abstract customer-service communication.</td><td>Network Service Layer: Contains services that abstract network resource and capability functions.</td></tr>
<tr><td>Connectivity Session: Components abstract network connectivity required for service.</td><td rowspan="4">Network Abstraction Layer: Contains services that abstract underlying telco network resources and capabilities. Provides a single and standardised point of integration between SDP services and the converged telco-IT networks.</td></tr>
<tr><td>Layer Network: Components abstract the network delivering the service.</td></tr>
<tr><td>Subnetwork: Components abstract network subnetworks.</td></tr>
<tr><td>Subnetwork: Components abstract subnetwork elements.</td></tr>
<tr><td>Network Element: Defines physical network elements used to provision services.</td><td>Network Element Layer: Translates higher layer technology independent requests to technology dependent operations on telco network elements and systems.</td></tr>
<tr><td>Distributed Domains</td><td colspan="3">Distributes layers and their contents across areas of operation, such as the customer domain. May be decomposed into additional domains.</td></tr>
<tr><td>Middleware</td><td>Abstracts distribution, technology and telco systems. Implemented is CORBA-based.</td><td>Middleware</td><td>Abstracts distribution, technology and telco systems. Represented as a technology neutral plane.</td></tr>
</table>

Table 4.2: Comparison of TINA and SDP concepts and architectures

To promote standardisation, service interfaces may be fully defined or implemented using existing service platform standards. For example, services contained within the generic and network service layers may implement their interfaces using Parlay, OMA or JAIN. However, services contained within the network abstraction layer may implement their interfaces using protocols such as SIP or SOAP [16].

The SDP middleware plane also represents a design pattern for structuring the SDP architecture. The middleware plane hides service and network distributions. Also, the middleware plane is technology independent and abstracts telco network technologies. Hence, the middleware plane ensures *communication* between services independent of their underlying computing technology, location and programming. In addition, the middleware plane provides services with interfaces that are used by SDP services, customers and providers. Some of these middleware services abstract underlying telco network systems, such as legacy service platforms, databases and other forms of middleware. In addition, the middleware services abstract management systems, such as the telco OSS/BSS. Therefore, by using standardised middleware plane service interfaces, the SDP architecture promotes standard-based interaction between telco OSS/BSS, SCE and SME.

4.3 Summary

In this chapter we presented contributions of legacy service platform standards to the SDP and its framework. We showed the IN conceptual model as providing a hierarchy of planes for the SDP framework. Planes separate and group specific services and network functions. These services and functions are abstractions that simplify access to the telco infrastructure. Also, abstractions provide technology neutral interfaces, enabling them to communication across planes. In addition, planes abstract distribution of communicating services and functions. We also applied TINA concepts to the SDP and defined a generic business model, that captures requirements and business domains. We decomposed the business model into a SDP architecture. The architecture used layers to separate and group SDP services. Services provide technology neutral interfaces to their functions. Layers and services are distributed across functional domains. The SDP architecture also used a middleware plane, containing a collection of services that abstract distribution, technology and telco network systems. Middleware services are used across all layers and domains via their technology neutral interfaces. Therefore, by reusing generic concepts from both IN and TINA we presented technology, implementation and distribution neutral abstractions and architectures that contribute to the SDP framework.

Chapter 5

Perspectives on the SDP from Service Platform Standards: Parlay and Parlay X

The IN is a legacy service platform standard that is deployed across many telco networks, while TINA represents a historical conceptual service platform standard that is not deployed. Currently, newer service platform standards contribute to the evolution of the telco network. These newer service platforms aim to manage and benefit from telecom-IT convergence. Popular examples of these service platforms are based on the Parlay [20] set of standards.

The Parlay group standardises a set of *Application Programming Interfaces (APIs)* that promote the opening of the telco network to external IT infrastructures. The APIs enable the full capabilities of the underlying telco network to be invoked using IT-based technologies and mechanisms. Hence, APIs enable application developers to create traditional telco voice-oriented services by exploiting telco network resources and capabilities [84].

With telecom-IT convergence, new telco network capabilities and resources are available. These new capabilities are offered by service platforms, media repositories and packet-based transport networks. As a result, the Parlay APIs further support application developers in the creation of communication, information and content-based services, that are offered to both consumers and enterprises [86]. Hence, Parlay supports the creation of single-media and multimedia-based services that are delivered across converged networks to a broad range of customers.

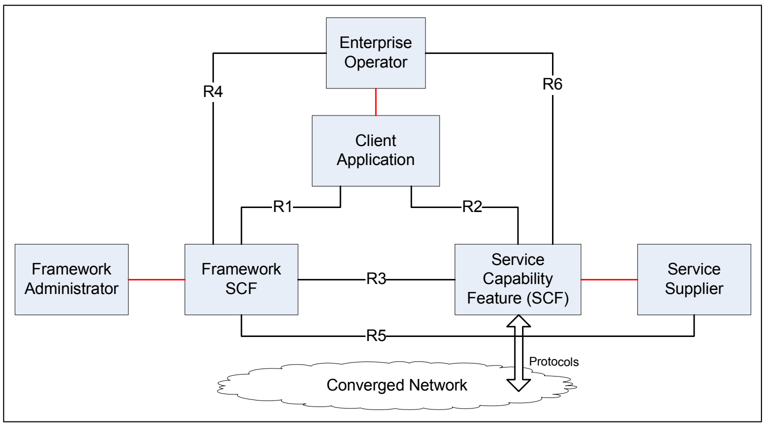


Figure 5.1: Parlay Reference Model

5.1 Requirements

As defined by [87], Parlay prescribes a variety of requirements for its APIs. For instance, APIs provide programmable access to all telco network functionality. Hence, software-based applications incorporate telco network functionality just as they would other forms of IT-based functionality. Also, APIs are implementable using diverse IT-based technologies, since they must be accessible by a variety of IT-using enterprises. The APIs abstract network functionality, such that telco network complexities are hidden from application developers. These complexities include network protocols, transport networks and distribution. In addition, the APIs provide an appropriate level of security since they open the telco network to external enterprises.

Based on these requirements, Parlay defines a reference model containing various interacting entities. The model is shown in *Figure 5.1*. In the figure, relationships between entities are defined, such that requirements are fulfilled. These relationships are specified using the standardised Parlay APIs. These reference model entities are:

- Enterprise Operator: represents an IT-using enterprise that is external to the telco. This entity requires access to telco network capabilities for its applications.
- Client Application: represents external IT-based applications that use telco network capabilities to provide services to customers. These applications may belong to the enterprise operator.
- Service Capability Feature (SCF): are software-based *services* that implement logic required to access underlying telco network capabilities.

- Framework SCF: represents a specialised SCF that provides administrative functions, such as SCF subscription, application authorisation and enterprise management. Also, provides broker functionality such as enabling applications to discover and locate other SCFs.
- Framework Administrator: represents a management entity that manages the framework SCF. This entity may be the telco or an external enterprise.
- Service Supplier: represents a type of management entity that develops SCFs and registers them with the framework, for use with applications. This entity may be the telco or an external enterprise.

The reference model interfaces are used to formalise entity relationships. For instance, interface $R1$ is used to authenticate client applications with the framework. Authorised applications access network capabilities via the SCFs, using interface $R2$. Interface $R3$ is used to locate and manage SCFs that have registered with the framework. This supports authorised client applications to locate SCFs via the framework. Using interface $R4$, enterprises register their applications with the framework, for use with SCFs. Also, enterprises use this interface to subscribe for SCFs that are used by their applications. The enterprise manages its SCF subscriptions via interface $R4$. By using interface $R5$, service suppliers register their SCFs with the framework. Interface $R6$ enables enterprise operators to invoke SCF capabilities.

In the reference model, interfaces $R5$ and $R6$ are to be standardised using the Parlay APIs. In addition, interfaces managing the administration of framework, applications and SCFs are not standardised. These administrative interfaces do not relate to the public telco network, but rather to private enterprises that administer applications and SCFs. Hence, these interfaces are not be managed by the Parlay APIs.

Based on the reference model a Parlay architecture is defined.

5.2 Architecture

The Parlay architecture defines and structures various abstractions that make use of the standard-based APIs. The main objective of the architecture is to promote the use of telco network capabilities by external IT-using enterprises, via the Parlay APIs. Also, the Parlay architecture provides a foundation on which other service-oriented standards are defined. In the following sections we discuss the Parlay architecture and its extension, the Parlay X architecture.

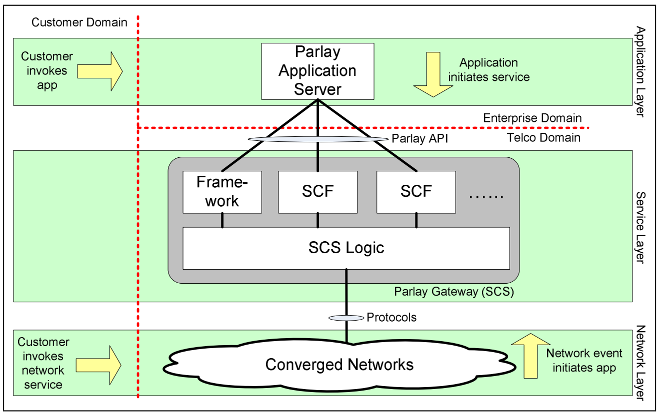


Figure 5.2: Parlay Architecture

5.2.1 Parlay

The Parlay architecture is illustrated in *Figure 5.2*, and is derived from [88, 89]. In the figure, the architecture contains a Parlay application server and Parlay gateway. The application server hosts various client applications and operates within the enterprise domain. Client applications access functionality offered by the Parlay gateway via the Parlay APIs.

The Parlay gateway is contained within the telco domain and is also known as the Service Capability Server (SCS). The gateway houses various SCFs that implement and expose the Parlay APIs. Examples of the SCFs and their associated APIs are:

- Framework SCF [90]: exposes the framework API to applications. Also, administers all other SCFs that have registered with it and performs various management and security functions.
- Generic Call Control SCF [91]: exposes the generic call control API, enabling applications to make, manipulate and manage calls within the telco network.
- User Interaction SCF [92]: exposes the user interaction API and enables applications to interact with customers using Interactive Voice Response (IVR) units, in a technology neutral manner.
- Mobility SCF [93]: exposes the mobility API, enabling applications to query the location of a customer's terminal.

Additional APIs are defined that enable applications to perform various functions, such as multiparty and multimedia call control, query customer terminal capabilities, control data sessions and manage customer accounts and presence information.

In *Figure 5.2*, the Parlay gateway (or SCS) contains software-based logic. When invoked by applications, SCFs invoke SCS logic interfaces to fulfil application requests. The SCS logic manages the interworking between SCFs and network specific functions. Hence, the SCS logic translates SCF requests into protocol messages that are used to invoke network functions. However, the SCS logic is unstructured and limited in standardisation. Examples of structured abstractions for the SCS logic are defined in [49, 94, 95].

In addition to the Parlay architecture, *Figure 5.2* illustrates how applications are invoked to provide customer services. First, on deployment applications may register with some SCFs for notification of specific network events. The network events may occur when a customer, within the customer domain, initiates a network service. When the event occurs, notifications are delivered to the appropriate SCFs, via the SCS. As a result, the SCF notifies the appropriate application to provide the customer service. Second, customers within the customer domain may directly access client applications and invoke a particular service. Once invoked, applications use SCFs to fulfill customer requests.

Parlay APIs are defined in a implementation neutral representation. As a result, APIs are implementable using a variety of technologies. Also, APIs are represented in a distribution independent manner, though they are invoked across enterprise and telco networks. Parlay promotes the use of standard-based technologies to implement its APIs. The use of standards ensure interoperability between different application and API implementations. In addition, the technologies must manage the distribution of the SCS, SCFs and applications. Parlay recommends CORBA-based technologies to implement APIs and hide implementation and distribution details.

Parlay APIs abstract functionality provided by underlying telco network protocols [96]. These APIs define abundant methods with rich data structures that are used by applications to invoke protocol functionality. As a result, Parlay-based applications that use the APIs have greater control over telco network functions. However, this implies application developers must have knowledge of telco network concepts to take full advantage of the APIs and the network capabilities [96]. Therefore, creating Parlay applications is complex due to the richness of the APIs and telco network knowledge that is required.

To simplify the complexity of the Parlay APIs, Parlay defines the Parlay X set of APIs.

5.2.2 Parlay X

IT-using enterprises require simple interfaces to access and use telco network capabilities. These interfaces must be void of any telco network details, such as protocols. As a result, Parlay defines the Parlay X APIs. These APIs simplify access and usage of the Parlay APIs and telco network functions. The Parlay X APIs are based on Internet technologies, called web services [7]. We discuss web services in *Chapter 7*.

Parlay X APIs specify fewer methods, simpler data structures and simpler interaction models than the Parlay APIs. The API methods expose the generic service-oriented functionality offered by the underlying telco network resources and capabilities. For example, the APIs provide methods to setup a simple two-party call using phone numbers that are managed within a simple data structure. As a result, the Parlay X APIs provide less control over telco network capabilities than the Parlay APIs.

Though less expressive than Parlay, the Parlay X APIs provide an acceptable level of abstraction for accessing network resources and capabilities. In addition, APIs are defined using Internet-based technologies. As a result, IT using enterprises and Internet-based application developers form the target market for Parlay X APIs [97]. The APIs enable developers to enhance their existing applications with telco network capabilities, or create new applications. Also, the APIs easily integrate with their existing IT infrastructure, such as their application platforms. Hence, Parlay X enables a diverse range of IT-based application developers to easily create applications, that benefit from using telco network resources and capabilities.

The Parlay X architecture is shown in *Figure 5.3*, and is derived from [98]. In the figure, the architecture consists of a Parlay X application server, Parlay X gateway and Parlay gateway. The Parlay X application server hosts various client applications and operates within the enterprise domain. These applications access functionality offered by the Parlay X gateway using the Parlay X APIs. The gateway is contained within the telco domain.

The Parlay X gateway houses various web services. Web services represent software-based logic that implement and expose the Parlay X APIs. Applications invoke web services via their interfaces to access network functions. Web services invoke Parlay APIs to fulfil application requests. Hence, Parlay X APIs simplify access and usage of the Parlay APIs. Numerous web services are defined to abstract the Parlay APIs. For example:

- Third party call web service [99]: enables external applications to initiate a call between customers, on the telco network. This web service simplifies the various Parlay call control APIs.
- Audio call web service [100]: enables applications to deliver audio to customers. This web service simplifies the Parlay user interaction API.
- Terminal location web service [101]: enables applications to obtain the location of

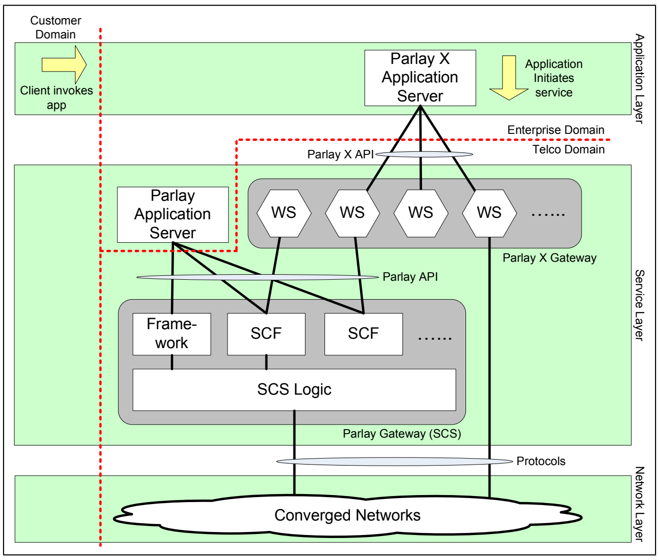


Figure 5.3: Parlay X Architecture

a customer's terminal. This web service simplifies the Parlay mobility management API.

Additional Parlay X APIs are defined and provide functionality such as, messaging, media streaming and presence management. However, Parlay X does not define a framework web service that abstracts the Parlay framework API. Since, Parlay X is based on web services, Internet-based mechanisms are proposed to provide framework-like functionality. One such mechanism is a Universal Description, Discovery and Integration (UDDI) [15] registry, that registers web services and enables applications to locate web services in a secure manner. We discuss these and other web service technologies in *Chapter 7*.

In *Figure 5.3*, web services are contained in the Parlay X gateway that interworks Parlay X applications and the Parlay SCFs. Therefore, web services may be considered Parlay client applications, since they invoke SCFs via the Parlay APIs. However, Parlay X standards also enable web services to directly use the converged network capabilities. This is only possible if the network resources and capabilities can directly communicate with the web services using Internet-based protocols. Some applicable network resources include the IT-based OSS/BSS.

In addition to the Parlay X architecture, *Figure 5.3* illustrates how Parlay X applications are invoked to provide customer services. Customers can access Parlay X applications using Internet-based technologies and invoke application functionality directly. Once invoked, the applications may use web services, via the Parlay X APIs, to access network capabilities to fulfil customer service requests. Also, web services may directly invoke the appropriate network functions to fulfil the customer request.

Parlay X APIs are defined in an implementation neutral representation. However, Parlay X promotes the use of web service technologies to implement the APIs. These technologies are discussed in *Chapter 7*.

5.3 Reusable Concepts

Parlay shares many concepts with the TINA service platform [87]. Some concepts include business modeling, abstraction through separation, standardised interfaces and hiding distribution complexities. These concepts are shared with Parlay X, since it is an extension of Parlay. As a result, both Parlay architectures provide generic and reusable concepts that benefit the SDP.

The Parlay reference model represents a business model containing interacting business entities using formalised relationships. In this business model, the framework SCF represents a service retailer that provides enterprise customers and their client applications with access to services (SCFs) [89]. The framework SCF also provides broker functionality since it enables enterprise customers to locate services. The reference model formalises business relationships between entities using specialised *reference points* called interfaces. The interfaces are implemented using the standardised Parlay APIs. The interfaces may also be implemented using the standard-based Parlay X APIs. For instance, web services represent services that are used by enterprise customers and their web-based client applications. However, Parlay X does not standardise a retailer, such as the Parlay framework.

Parlay uses the concept of *separation* to define its architectures. Both Parlay and Parlay X separate service-oriented capabilities from the underlying telco network into generic *services*. These services are the SCFs and web services. Services provide access to network capabilities by implementing the standard-based Parlay and Parlay X APIs. These APIs represent the decomposition of interfaces $R1$, $R2$, $R3$ and $R4$ of the Parlay reference model, shown in *Figure 5.1*. However, both Parlay and Parlay X APIs do not implement administration interfaces, that are used by the framework administrator and service supplier. These interfaces enable management applications to administer Parlay and Parlay X gateways. These administration interfaces require standardisation.

Parlay and Parlay X APIs are implemented and exposed by SCF and web services respectively. Web services manage the translation between Parlay X and Parlay API invocations and between Parlay X API invocations and Internet-based protocols. Hence, web services provide a *point of integration* between applications, Parlay and network functions. Parlay SCFs manage the translation between Parlay API invocations and underlying converged network protocols. However, the SCFs use the SCS logic to manage the interworking with the underlying network. Hence, the combination of SCFs and SCS logic provides a *point of integration* between applications (including web services) and network protocols.

Both Parlay and Parlay X APIs are implementation *independent*. As a result, various programming languages may be used to implement the APIs, SCFs, web services and applications. The APIs are also distribution independent. As a result, implemented SCFs and web services may be distributed across service platforms and the telco network, but remain accessible to applications. Hence, applications may invoke Parlay or Parlay X APIs independent of their implementation or location.

In both Parlay architectures applications, web services, SCFs and the network are modeled as separate horizontal layers. Layers are structured hierarchically with Parlay X forming the topmost layer, Parlay forming the middle layer and the network forming the bottommost layer. Each layer simplifies lower layers, such that the complexity of the underlying network protocols are abstracted. Layers' services vertically communicate via the Parlay and Parlay X APIs. Service communication also occurs horizontally within layers, via service interfaces. For example, client applications may be invoked by customer applications, across the application layer.

Though not formalised in the Parlay standards, the Parlay architectures operate across various domains. These domains include the enterprise domain, where client applications reside. The enterprise domain is separate from the telco since it contains the enterprise operators infrastructure. A telco domain is also defined and contains the Parlay and Parlay X gateways. In addition, the telco domain contains the network resources and capabilities. Customers access applications within the customer domain. The customer domain may include individuals, enterprises or other telcos.

SCFs, web services and applications operate across multiple domains. Domains imply a distributed environment that SCFs, web services and applications must operate across. As a result, Parlay promotes the use of distributed processing technologies to abstract the complexity of distribution. These technologies provide software-based mechanisms to hide distribution and implementation of services and applications from each other. Hence, these technologies enable service and application implementations to communicate, in a distribution and technology independent manner. We generalise the distributed processing technologies as a *middleware plane*.

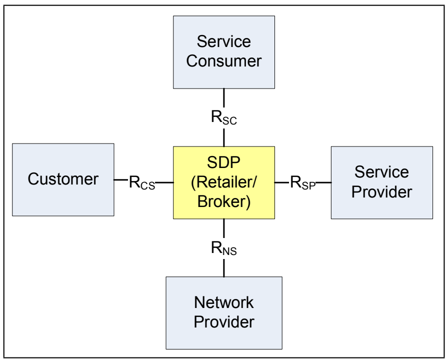


Figure 5.4: Generic SDP Business Model derived from Parlay Reference Model

5.4 Contribution to the SDP from Parlay and Parlay X

We reuse and extend the generic and technology neutral Parlay and Parlay X concepts to define a SDP business model, reference model and architecture.

The SDP business model is illustrated in *Figure 5.4* and is derived from the Parlay reference model shown in *Figure 5.1*. The model is abstract and illustrates the SDP performing the business roles of a service retailer and service broker. As a result, the SDP provides services to various business entities. Also, business entities use the SDP to locate services. The entities that interact with the SDP include:

- Service Providers: perform OSS/BSS specific functions with the SDP. For instance, service providers use SCE and SME to create and manage their own services, that are integrated into the SDP and contribute to the SDP's service repository. This business entity is similar to the service supplier in the Parlay reference model.
- Service Consumers: represent application providers who create applications by locating and consuming specific services offered by the SDP. Consumers orchestrate services into their applications that are used by customers. This business entity is similar to the enterprise operator and client applications in the Parlay reference model.
- Customers: use SDP services to locate and access applications. These applications provide customer services. This entity is lacking in the Parlay reference model.
- Network Provider: provides the underlying network resources and capabilities required by SDP services to support application development and customer service deliver. This business entity is abstracted by the SCFs in the Parlay reference model. Also, the Parlay reference model does promote a standardised interface between the SCFs and this business entity.

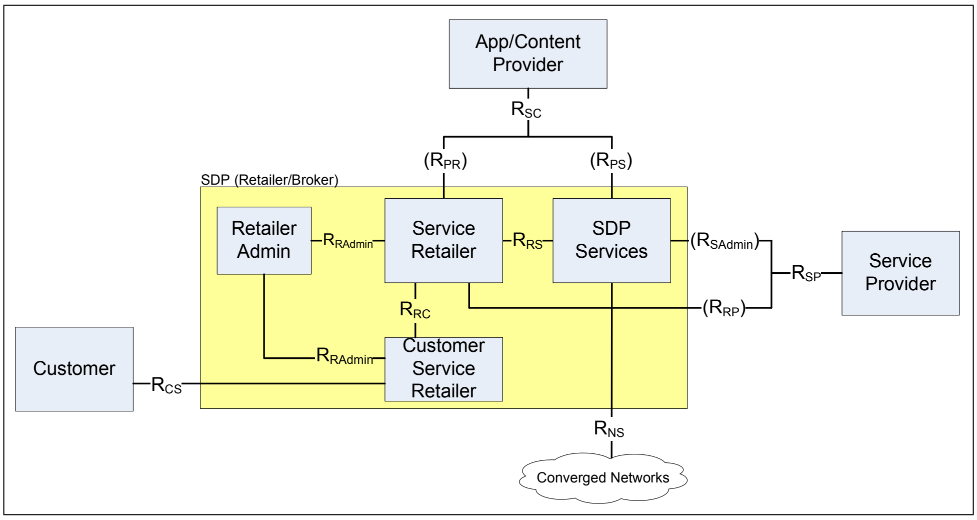


Figure 5.5: SDP Reference Model derived from Parlay Reference Model

Business interactions between entities in the SDP business model are defined using business relationships. These relationships prescribe rules and contracts between the SDP and business entities. For instance, the R_{SC} relationship defines access and usage permissions of services by service consumers. The R_{SP} relationship specifies rules for service providers to deploy their services using the SDP. In addition, the R_{NS} relationship regulates use of network resources and capabilities by the SDP and services. Also, the R_{CS} relationship defines usability of the SDP by customers.

The SDP business model is decomposable into a reference model. An example reference model is shown in *Figure 5.5*. The reference model is also derived from the Parlay reference model shown in *Figure 5.1*. The reference model decomposes the SDP business model entity into additional components, that support the business relationships. These components include:

- Service Retailer: represents a component of the SDP business entity. This component provides services to service consumers' applications. Thus, it provides functionality similar to the Parlay framework SCF. In the SDP reference model we show service consumers as application or content providers.

- SDP Services: represent a diverse range of services that are created and managed by service providers. Some SDP services may provide service consumers with simple access to network capabilities, while others provide services to customers. Thus, SDP services represent abstractions of Parlay SCFs and Parlay X web services.

- Customer Service Retailer: enables customers to locate and use customer services, via the service retailer. These customers are not enterprise operators or their client application, but are similar to end-users of applications or TINA consumers.

- Retailer administrator: provides administrative functionality to manage the SDP retailers. This entity is similar to the Parlay reference model's framework administrator.

In the reference model, relationships between SDP business entity components are standardised as *reference points* before they are implemented using standard-based APIs. For example, the R_{RS} reference point promotes standardised interactions between the service retailer and SDP services. These interactions enable the service retailer to locate SDP services for service consumers. This reference point relates to the $R3$ interface of the Parlay reference model. The R_{RC} reference point standardises communication between the service retailer and customer service retailer. This communication enables the customer service retailer to locate customer services. There are no interfaces in the Parlay reference model that corresponds to this reference point. The R_{Admin} enables the SDP to manage its retailers. This reference point relates to the nonstandardised interface between the framework administrator and framework SCF in the Parlay reference model.

Business relationships between SDP business entity components and external business entities are also formalised as reference points. These reference points share names with the business relationships, that is, R_{SC}, R_{SP}, R_{NS} and R_{CS}. These reference points promote standard-based communication between the SDP and external entities, including the converged networks and IT-using enterprises. However, the R_{SC} and R_{SP} reference points are decomposed into more specific reference points, since they interact with the SDP components. The R_{SC} business relationship is divided into the R_{PR} and R_{PS} reference points. The R_{PR} reference point standardises interactions between service consumers and the service retailer. These interactions enable consumers to locate and register to use SDP services. Also, consumers may register their applications for use by customers. This reference point corresponds to the Parlay reference model's $R1$ and $R4$ interfaces. The R_{PS} reference point standardises interactions between service consumers, their applications and SDP services. Hence, this reference point standardises usage of SDP services and therefore, the underlying network capabilities. This reference point relates to the Parlay reference model's $R2$ and $R6$ interfaces.

The R_{SP} reference point is divided into the R_{RP} and R_{SAdmin} reference points. The R_{RP} standardises communication between the service retailer an service provider. As a result, diverse service providers can register their services with the service retailer, in a consistent manner. This reference point is similar to the $R5$ interface of the Parlay reference model. The R_{SAdmin} standardises interactions between the service supplier and their services. Hence, service providers manage their services, while they are deployed in the SDP. This reference point relates to the nonstandardised interface between the service supplier and SCFs in the Parlay reference model. The R_{CS} standardises interactions between customers and the customer service retailer. These interactions enable customers to locate and register to use applications that provide specific customer services. This reference point has no corresponding interface in the Parlay reference model.

Like Parlay, the SDP reference points in *Figure 5.5* are implemented as generic interfaces. Interfaces express functionality offered by the business entity components. For example, SDP services offer external IT-using enterprises interfaces, that expose the service-oriented capabilities of the underlying network. Also, these SDP service interfaces express functionality independent of telco details, such as protocols. Other interfaces describe functionality to support communication between business entity components. All interfaces are defined in an implementation neutral manner, such that they are implementable using a variety of technologies.

All interfaces are implemented and exposed using services. As a result, each business entity component contains services that implement and expose their appropriate interfaces. For example, the service retailer contains retailer services that enable application developers to locate SDP services. Also, the customer service retailer contains its own retailer services that enable it to locate, subscribe and access applications, on behalf of customers. The SDP service entity contains a diverse collection of services. These services are of the following types:

- Application services: expose simple interfaces to service consumers to access network capabilities. However, some application services may not expose interfaces but provide services to customers.
- Generic services: provide interfaces for application services to invoke the service-oriented capabilities of the underlying network resources.
- Network services: provide interfaces for generic services to invoke network functions independent of their technologies.

Some of the above SDP services may be created and managed by external service providers. For example, *Figure 5.5* shows a service provider interacting with the SDP to deploy and manage its own services.

Services and their interfaces implementing the SDP reference model are managed in a SDP architecture illustrated in *Figure 5.6*. The SDP architecture is derived from the Parlay and Parlay X architectures shown in *Figure 5.2* and *Figure 5.3*. The architecture represents the SDP within an environment of entities that invoke SDP service interfaces.

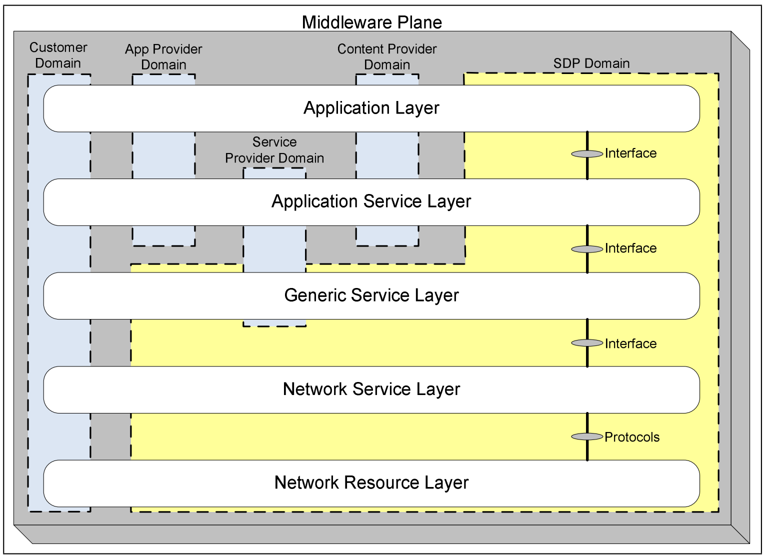


Figure 5.6: SDP and its Environment derived from Parlay and Parlay X Architectures

The SDP architecture separates services into horizontal layers that are hierarchically organised based on their level of abstraction. The application layer contains applications that consume application services. These applications may belong to application or content providers The application service layer houses the various types of application services. These services may be implemented as Parlay SCFs or Parlay X web services. The generic service layer contains generic services. These services may be implemented as Parlay SCFs. The network service layer groups network services. The network services may be implemented as part of the Parlay SCS logic. The network resource layer houses the converged network resources and capabilities, such as transport networks, OSS/BSS and data stores. These resources are invoked by the network services. All service layers expose their service interfaces to layers higher in the SDP architecture.

If Parlay X web services are used to implement the application service layer, we may remove the generic service and network service layers. As a result, the web services can communicate directly with the appropriate network resources. Thus, the SDP architecture can be reduced to accommodate the Parlay X architecture shown in *Figure 5.3*.

In addition to layers, the SDP architecture defines domains. Domains represent functional divisions across the SDP architecture. The SDP architecture defines domains that correspond to the business entities that interact with the SDP business entity and components. Hence, services derived from the business entities operate within their associated domains. However, SDP services are accessed and used across multiple domains. SDP services provide their functionality across various domains by using their interfaces. For example, *Figure 5.6* shows application services are accessible via their interfaces, by applications contained in the application provider or content provider domains. Also, service providers access generic service, via their interfaces, from within their domain when creating and testing their own application services.

Like Parlay and Parlay X, middleware is essential to the SDP architecture. The SDP uses middleware mechanisms to abstract the distribution of applications and services across various domains. The middleware mechanisms also contain functionality to support implementation and distribution independence of service interfaces. The SDP middleware abstracts underlying technology details, such as computing platforms that host applications and services. Also, the middleware abstracts transport networks used to deliver application and service communications. In the SDP architecture middleware mechanisms are abstracted as a middleware plane. In *Figure 5.6*, the middleware plane encapsulates all layers and domains, such that implementation, distribution and technology details are abstracted.

5.5 Evaluation of SDP Business Model, Reference Model and Architecture

By applying the generic Parlay and Parlay X concepts to the SDP we have defined a telecom service-oriented viewpoint. This viewpoint defines various abstractions that are structured in the SDP business model, reference model and architecture. The abstractions contribute to the definition of the SDP framework. We evaluate the SDP business model, reference model and architecture against the generic concepts extracted from Parlay and Parlay X. In addition, the evaluation provides answers to the questions posed in *Chapter 1 Section 1.3*. We also present the comparison between Parlay, Parlay X and SDP in *Table 5.1*.

The SDP business model justifies the development of the SDP by a telco. That is, the SDP manages and benefits from telecom-IT convergence, by providing diverse services to external IT-using enterprises. The business model remains generic so as to cater for many types of enterprises (business entities). Interactions between SDP and enterprises define business relationships and their roles in these relationships. The most predominant role between SDP and enterprises is that of service *provider* and service *consumer* respectively. However, the business model also shows the SDP as a service consumer that deploys services on behalf of external service providers.

The SDP reference model adds detail on the business model, such as the internal components that constitute the SDP business entity. Also, the reference model shows examples of enterprises that interact with the SDP components. Some of these enterprises are service consumers, such as application developers, content providers and customers. In addition

<table>
<tr><th colspan="2">Parlay X</th><th>Parlay</th><th>SDP</th><th>Description</th></tr>
<tr><td colspan="3" rowspan="2">Reference Model</td><td>Business model</td><td>Defines telco and IT-based business entities and the business relationships between them.</td></tr>
<tr><td>Reference Model</td><td>Decomposes business entities and formalises business relationships as reference points.</td></tr>
<tr><td colspan="2">Web Application Layer</td><td></td><td>Application Layer</td><td>Contains IT-based applications that access network capabilities to provide customer services.</td></tr>
<tr><td rowspan="3">Web Services Layer</td><td colspan="2">CORBA Application Layer</td><td>Application Service Layer</td><td>Defines simple services that provide access to network capabilities, via technology neutral interfaces. Some application services may not expose interfaces, but provide customer services.</td></tr>
<tr><td colspan="2" rowspan="2">Service Capability Function Layer</td><td>Generic Service Layer</td><td>Contains services with interfaces that simplify access to the service-oriented capabilities of the underlying network.</td></tr>
<tr><td>Network Service Layer</td><td>Contains services with interfaces that promote access to the underlying network functions.</td></tr>
<tr><td colspan="4">Network Resource Layer (telco, IT-using enterprise and Internet)</td><td>Represents the integrated collection of network resources and capabilities that are technology, implementation and distribution specific.</td></tr>
<tr><td colspan="4">Distributed Domains</td><td>Represent areas of division across the SDP architecture. Domains relate to business entities found in the business/reference model.</td></tr>
<tr><td>Protocols</td><td colspan="2">CORBA</td><td>Middleware Plane</td><td>Supports application and service communication by abstracting their implementation, distribution and technology details.</td></tr>
</table>

Table 5.1: Comparison of Parlay and SDP Concepts and Architectures

to service consumers, the reference model shows service providers, who deploy their services with the SDP. The reference model promotes *standardised* interactions between SDP components, external enterprises and network using reference points. The reference model reference points are implementable as service interfaces. Thus, the SDP and enterprises implement reference points using numerous services that expose specific interfaces. As a result, interfaces expose SDP capabilities to enterprises in a standardised manner, but, ensure the inner-workings of the SDP components are hidden. Service interfaces express SDP capabilities independently of network technologies, making them easier to use when programming new applications. Also, service interfaces are implementation and distribution neutral, such that diverse application implementations may use their functions across various locations. Thus, the abundant SDP services and their interfaces provide varying levels of abstraction of the telco network resources and capabilities.

The SDP services and interfaces are structured within an architecture using layer, domain and plane design patterns. Unlike the proposed SDP architecture shown in *Figure 1.3*, this

SDP architecture is technology neutral and uses service interfaces to promote SDP standardisation. These SDP architecture layers and groups services as a hierarchy of abstractions that simplify access and usage of the underlying network resources. The various service interfaces enable standardised communication within and across layers. Also, services and their interfaces enable consistent access to the SDP and the underlying network capabilities. For example, application services provide a *point of integration* between external applications and generic services. The network services provide a *point of integration* between generic services and the converged network resources. The generic services translate between application service invocations and network service invocations. Some SDP services also provide a means for customers to locate applications that provide diverse functionality. Hence, these SDP services support a standardised *point of integration* between customers and external applications. The SDP service interfaces ensure these points of integration remain technology neutral, by hiding details of the service implementation, distribution and underlying network technologies.

The SDP architecture represents a distributed platform, since its layers operate across multiple domains. Services are distributed across these domains, but remain within their associated layers. Service interfaces enable their distribution to be abstracted, to some degree. However, the *middleware* plane design pattern enables service distribution to be fully abstracted. Also, the middleware plane provides functionality to abstract application and service implementations and distribution. For example, the middleware plane abstracts diverse:

- computing platforms that host applications and services;
- software implementations of applications and services; and
- transport links used to deliver application and service communications.

As a result, the middleware plane abstracts various complexities, such that application development and service execution is simplified. Thus, the middleware plane provides a point of integration between applications, services and their networked computing platforms. The SDP architecture defines the middleware plane independently of technologies. Hence, the middleware plane may be implemented using a single technology or an integration of many technologies.

Both Parlay and Parlay X do not provide standardised APIs that completely abstract telco OSS/BSS capabilities. This is also evident in the Parlay reference model's nonstandardised interfaces involving the framework administrator and service supplier. As a result, our derived SDP business model, reference model and architecture does not incorporate the telco OSS/BSS or external enterprises' SCE and SME. These issues are resolved in chapter *Chapter 6*.

5.6 Summary

In this chapter we presented contributions of current service platform standards to the SDP and its framework. We reused generic concepts from both Parlay and Parlay X standards. These concepts where extracted from the Parlay reference model and Parlay and Parlay X architectures. By reusing concepts from the Parlay reference model we defined a SDP business model that motivates the business case for the SDP. Also, we decomposed the business model into a SDP reference model with reference points, to promote SDP standardisation. Like Parlay and Parlay X, we reused the concept of standard-based service interfaces to implement reference points. For example, service interfaces hide the complexity of network technologies from applications. Also, the service interfaces are implementation and distribution independent. We decomposed the reference model into a SDP architecture, that grouped and structured SDP services into horizontal layers. Also, we used domains to distributed the layers across various functional areas. We incorporated a technology neutral middleware plane to hide implementation, distribution and technology complexities from services and applications. Therefore, by reusing generic Parlay concepts we presented technology neutral abstractions and an architecture that contributes to the SDP framework.

Chapter 6

Perspectives on the SDP from Management Framework Standards: TMN, TOM and eTOM

Telco network infrastructure contains varied integrated systems. Each system is distributed across the telco network and uses diverse technologies to interoperate with each other. The networked collection of systems support the telco business, that is, the development and delivery of customer services. The telco uses specialised systems to manage its infrastructure and customer services. These systems are the operations and business support systems (OSS/BSS).

OSS/BSS form a crucial part of the telco infrastructure. OSS ensures the effective operation of the telco infrastructure parts, such as switches, routers, databases, service platforms and customer terminals. The BSS manages telco business objectives by defining and implementing business processes. For instance, business processes define logic to deploy services, perform network configuration and bill customers. The OSS and BSS are integrated such that the operation of the telco infrastructure satisfies business objectives.

With convergence effecting both telco business and infrastructure, its traditional OSS/BSS is changing. The OSS must manage integration of legacy telco systems with new telco, IT and Internet-based systems. The OSS must manage the opening of the telco infrastructure to various external partners. Also, the OSS must manage old and new types of services that are defined, deployed and maintained by the telco. In relation, the BSS must manage existing business processes, but support new processes that enable the telco to benefit from telco, IT and Internet convergence. For example, telco business processes must satisfy business objectives that include external partners, such as application developers, content providers and other telcos.

Various tools are defined to aid development of a telco OSS/BSS that manages telecom-IT convergence. In this chapter we discuss tools, such as the Telecommunication Management Network (TMN) [21], Telecommunication Operations Map (TOM) [22] and Enhanced Telecommunication Operations Map (eTOM) [23]. These standards provide a wealth of generic and reusable concepts that focus on the operations and business management of telco infrastructure. Hence, these concepts aid the development of the SDP framework, since the SDP must manage telecom-IT convergence.

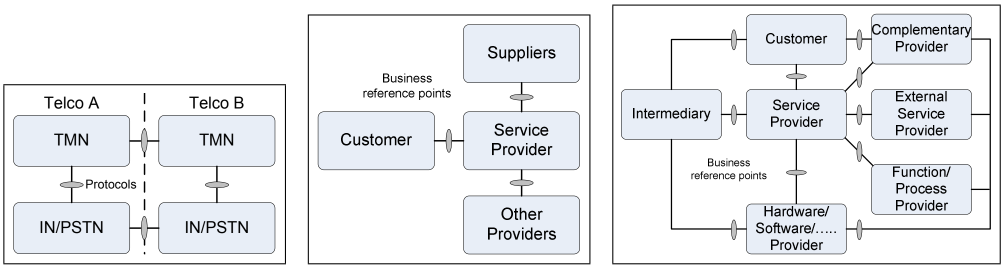
6.1 Requirements

6.1.1 TMN

Legacy telco infrastructure provides voice services to customers over limited transport networks. For example, the IN is used to create and execute voice-based services, while the PSTN is used to deliver IN services across circuit-mode transport networks. To manage legacy telco infrastructure the TMN is defined. The TMN represents a separate and dedicated management network that integrates with the IN/PSTN and other TMNs, to provide various management functions [102, 21]. As defined in [103], TMN functions aim to fulfil the following requirements:

- Fault Management: detect, isolate, report and correct errors found in telco infrastructure.
- Configuration Management: adjust properties of telco infrastructure systems, so as to control their operation.
- Accounting Management: collect diverse information, based on service/network usage by customers, and adjust billing records.
- Performance Management: monitor and adjust telco infrastructure system properties, such that they operate efficiently.
- Security Management: enables the creation, modification and application of various security policies that ensure authorised access to management functions and information.

We define a TMN business model, shown in *Figure 6.1(a)*, that summarises TMN requirements. The business model illustrates separate TMNs managing separate networks using standard-based protocols. In addition, standard-based protocols are used to integrate both networks and their corresponding TMNs.



(a) TMN Business Model (b) TOM Business Model (c) eTOM Value Network

Figure 6.1: Telco Management Business Models

6.1.2 TOM

The TMN represents a legacy approach to define a telco OSS/BSS. The TMN aims to satisfy limited OSS/BSS requirements by defining network-oriented services, functions and protocols. In addition, the TMN provides incomplete standardisation, with implementations using proprietary mechanisms [104]. To overcome these limitations, the TOM is defined.

TOM represents a standardised business process framework that is used to develop telco OSS/BSS. TOM defines various business processes, with each describing activities that a telco OSS/BSS must support. These business processes are highly abstract and encapsulate the telco OSS/BSS requirements. By linking various business processes together, specific telco business objectives are satisfied. Consequently, the fulfillment of business processes influence the operation of the telco infrastructure. Based on [22], TOM and its business processes fulfil requirements, such as:

- End-to-end automation of business and operation activities, for example, automation of service creation, deployment, delivery and billing.
- Promote customer and service centricity, rather than network centricity. This enables business processes to focus on telco OSS/BSS requirements, rather than the underlying network technologies.
- Incorporate external partners into the telco business model, such that partners use services offered by the telco. Partners perceive the telco as a service provider. For example, partners use telco services to *supply* their own services to customers. In addition, some partners use telco services to *provide* content to enhance customer services.
- Administer distributed and diverse management information, obtained from heterogenous telco infrastructure systems.
- Reuse legacy OSS/BSS capabilities, since they manage existing telco infrastructure

systems.

- Ensure interoperability between different telco OSS/BSS implementations.

The TOM requirements are summarised in a business model that is defined in [22] and shown in *Figure 6.1(b).* The model defines the telco service provider, customers, suppliers and other provider business entities. Examples of suppliers include application developers, while other providers include content providers. All business entities are involved in various business relationships, called business reference points. Business reference points promote standardised management interactions between telco, customers, suppliers and providers. These interactions contribute to the definition of the TOM business processes.

6.1.3 eTOM

The eTOM is defined due to refinement of the TOM standard. The eTOM shares properties, requirements and business processes with TOM. However, eTOM enhances TOM by managing the effects of telecom, IT and Internet convergence on the telco OSS/BSS. As a result, eTOM defines additional business processes that manage the integration of telco OSS/BSS with external IT-based enterprise infrastructures and the Internet. In addition, eTOM defines business processes that promote improvement of internal telco activities. Hence, eTOM provides a more complete approach to develop telco OSS/BSS.

The eTOM business model is defined in [23] and shown in *Figure 6.1(c).* The business model is generic and illustrates a network of telco, enterprises and customers involved in various business relationships, called business reference points. Like TOM, these business reference points promote standardised interactions between business entities, that contribute towards the definition of eTOM business processes. In the figure, the business entities and their functions are:

- Service Provider: represents a telco that purchases products or services from various entities in the business model. Also, the telco sells services to entities in the business model.
- Customer: is an individual or enterprise that subscribes, consumes and pays for customer services.
- Hardware/Software Provider: sells products to the business model's providers. These products are then integrated into the providers infrastructure.
- External Service Provider: is an external telco or enterprise that uses the service provider's services to create, execute, enhance, deliver or manage its own customer

services. In addition, the service provider may use external service providers' services to create, execute, enhance, deliver or manage its own customer services.

- Complimentary Provider: represents an external enterprise that adds value to the service provider's customer services. For example, this provider may represent a media broadcaster who offers content to the service provider, for use in new customer services.

- Function/Process Provider: represents an enterprise that performs functions or even business processes on behalf of the service provider. For example, a function may include selling customer services to customers.

- Intermediary: is an enterprise that takes on the role of a broker. For instance, the broker enables business entities to locate each other for the purpose of buying or selling services, products or content.

eTOM is used in conjunction with other standard and technology neutral tools to aid the development of telco OSS/BSS. These tools include the Shared Information and Data Model (SID) [105] and Technology Neutral Architecture (TNA) [106]. The collection of these tools form part of a OSS/BSS development life cycle named the New Generation Operations Systems and Software (NGOSS) [48].

6.2 Architecture

6.2.1 TMN

The TMN functional and logical architecture is defined in [21] and shown in *Figure 6.2*. The architecture uses management layers to group functional entities based on their management capabilities. All lower layer functions are accessed and used by higher layer functions. The business management layer groups business Operations System Functions (OSF) that manage the telco business as a whole. This function obtains simple management information and uses management capabilities from lower layers to satisfy telco business objectives. The service management layer contains a service OSF that administers service oriented aspects of the telco infrastructure. For example, the function provisions IN-based services for customers, by using lower network functions.

The network management layer houses a network OSF that provides a holistic view of the telco network and abstracts details such as subnetworks and network elements. For example, the network OSF is used to configure the network when a new service is being deployed.

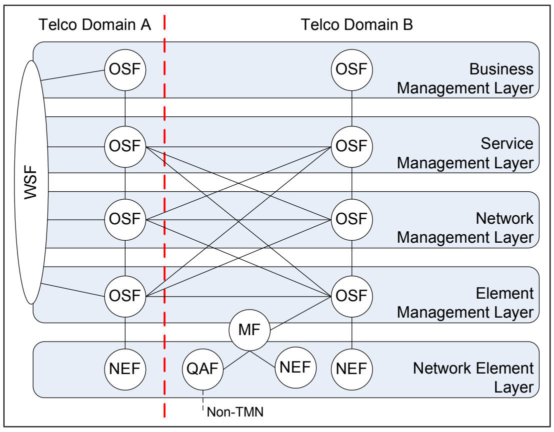


Figure 6.2: TMN Logical and Functional Architecture

The element management layer provides an element OSF that fulfills network administration by managing one or more network elements. The network element layer contains various functions that abstract access to a network element's management capabilities. For instance, Network Element Functions (NEF) abstract network element management capabilities, while Q-Adaptor Functions (QAF) abstract functions or network elements that do not adhere to TMN standards. In addition, Mediation Functions (MF) are used by element OSFs to abstract information obtained from multiple QAFs and/or NEFs. In addition, MFs may be used, throughout the TMN architecture, by higher layer OSFs to abstract information from multiple lower layer functions.

The TMN architecture shows interworking of TMN-compliant telcos. This interworking occurs across distributed telco infrastructure. TMN recommends interworking occur between service, network and element management layers, via the corresponding OSFs. Thought not depicted in the figure, the TMN architecture formalises all functional entity communication using reference points. Reference points enforce TMN standardisation by specifying the details of entity communication using protocols. For instance, the complex standard-based Common Management Information Protocol (CMIP) [107] is used to transport management information between implemented management functions. As a result, conformance to protocol specifications ensures interoperability between TMN implementations.

6.2.2 TOM

Unlike TMN, TOM represents a business process framework, rather than a management network architecture. The TOM framework structures business processes required by a telco to satisfy its business objectives. The TOM framework is defined in [22] and shown in *Figure 6.3*.

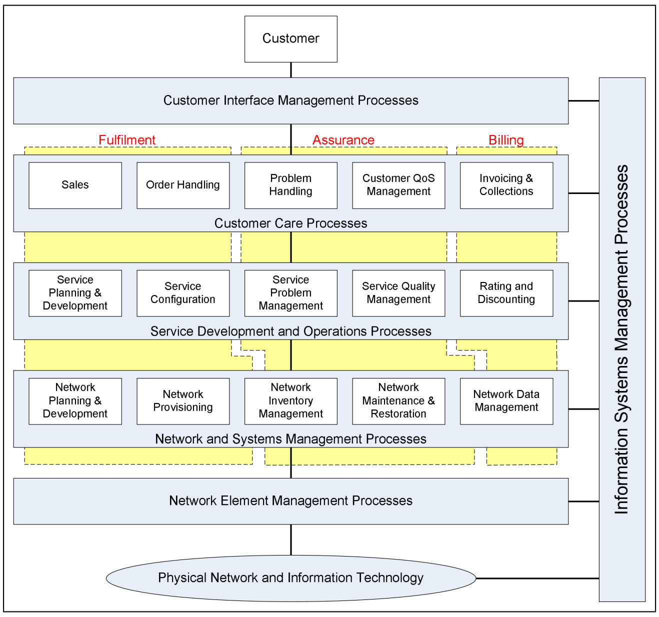


Figure 6.3: TOM Business Process Framework

Various business processes are defined in TOM. These business processes are grouped within functional layers that relate to the TMN management layers [108]. The customer care layer contains processes that aim to fulfil customer-oriented requests. This layer is similar to the TMN service management layer. Access to processes in the customer care layer are managed via a customer interface, such as a web portal. As a result, a customer interface management layer is defined. This layer houses business processes that administer the customer interface. The TOM also uses a service development and operations layer to house service-oriented business processes. These processes enable telco service management, independent of underlying telco network details. This layer also provides functionality similar to the TMN service management layer.

The network and systems management layer contains business processes that are used to satisfy the functions contained within the service development and operations process layer. This layer and its processes abstract the underlying complexity of network elements and their rich management information. This layer provides functionality similar to the TMN network management layer. To fulfil the network and systems management processes, network element management processes are defined. These processes are managed and grouped within the network element management layer. Like the TMN element management layer, the TOM network element management layer provides a consistent management interface to the underlying network elements. Integrated into all layers is the information systems management processes. These processes are not fully defined by TOM, but aim to manage the various information systems that implement the TOM business processes.

Each layer and their business processes contribute to one or more end-to-end process flows.

Process flows prescribe the linked execution of business processes across layers to fulfil business, service and customer requirements. We describe these process flows as end-to-end domains in the TOM. TOM defines three service-oriented domains, that is, service fulfillment, assurance or billing. The service fulfillment domain groups business processes that are used to support the delivery of services to customers. Service assurance includes business processes that maintain the appropriate levels of service reliability, performance and security. The service billing domain includes business processes that manage all aspects of service charging and customer billing. As shown in *Figure 6.3*, some business processes may overlap across two domains. Hence, some business processes are used in more than one end-to-end process flow.

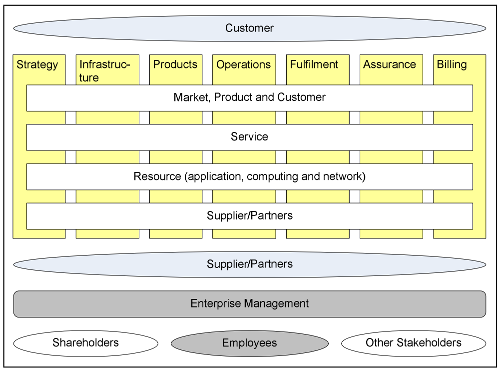
TOM defines interactions between business processes based on inputs to a business process and the corresponding output. An output for a business process may represent and input to another business process. A business process may invoke another business process:

- horizontally within the same layer and domain;
- horizontally within the same layer, but across different domains;
- vertically across different layers, but within the same domain; or
- vertically across different layers and domains.

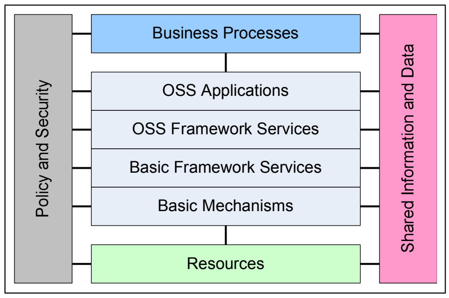
6.2.3 eTOM

Like TOM, the eTOM defines a framework of business processes that aim to satisfy telco business objectives. These objectives are customer and service oriented. Also, eTOM aims to enhance telco OSS/BSS to benefit from telecom, IT and Internet convergence. As a result, eTOM defines additional business processes, layers and process flow domains. The eTOM framework is defined in [22] and shown in *Figure 6.4(a)*. This figure illustrates an abstract version of the eTOM framework. More decomposed representations of eTOM and its business processes are defined in [23].

Like TOM, eTOM groups business processes into layers, according to their management capabilities. The market, product and customer layer contains business processes that provide marketing management and customer relationship management functions. The services layer houses business process that provide service-oriented management capabilities, such as service configuration. The resource layer groups business processes that administer the telco's diverse infrastructure. For example, resource business processes are used to configure transport networks. The supplier and partner layer contains business processes



(a) eTOM Level 0 Business Process Framework



(b) TNA Model

Figure 6.4: Instances of the eTOM Framework and TNA

that manage interactions with telco partners, such as content providers or other telcos. The eTOM also defines an enterprise management layer. This layer contains business processes that manage the internal activities of the telco. These processes are used or influenced by the telco employees, shareholders and stakeholders.

Similar to TOM, eTOM business processes contribute to one or more end-to-end process flows that are grouped within domains. These domains are distributed across all eTOM layers. eTOM defines a strategy domain that links and executes business processes used in defining and tracking the telco business strategies. The infrastructure domain involves business processes required to manage the life cycle of telco infrastructure. The product domain uses business processes required to manage the life cycle of telco services. Like TOM, eTOM reuses the service fulfillment, assurance and billing domains. However, eTOM defines an additional operations domain containing business processes, that support the service fulfillment, assurance and billing domains business processes. For example, the operations domain business processes provide information required by the service billing domain business processes.

As part of the NGOSS, eTOM represents an important tool in defining a telco OSS/BSS. The eTOM prescribes the business processes required to fulfil telco business objectives. These business processes are described at a high level of abstraction and encompass customer and service-oriented requirements. However, other tools are used within the NGOSS to transform the eTOM business processes into the implemented telco OSS/BSS. These tools are the SID and TNA. SID is used to model vital information required by the eTOM business processes [109]. For example, interacting business processes are decomposed into information processing objects. These objects properties are also described in detail using SID. In addition, SID specifies object interactions using information flows.

Once defined, the eTOM and SID models are used to aid the definition of the telco OSS/BSS architecture. This architecture is the TNA. The TNA is used to design and structure the telco OSS/BSS independently of technologies [106]. Also, the TNA structures the OSS/BSS as a distributed system. A TNA representation is defined by [110, 111] and shown in *Figure 6.4(b).* The architecture shows the basic components that constitute the structure of an NGOSS compliant OSS/BSS. The architecture uses layers to group various capabilities and functions. The topmost layer contains the eTOM business processes that are used to satisfy business objectives. The OSS application layer offers business processes access to legacy and new telco OSS/BSS capabilities. The OSS applications fulfil their requirements by using services offered by a OSS framework. The OSS framework is structured using two layers, that is, the OSS framework services layer and basic framework services layer. The OSS framework layer provides OSS specific services such as logging, trading and auditing services to OSS applications [112]. The basic framework services layer contains basic capabilities required to support OSS framework services, such as registration, naming and location services [112]. OSS applications may invoke either framework layers' services.

Since the TNA is a distributed system, capabilities are required to abstract distribution of business processes and framework services. Providing the distribution independence is the basic mechanism layer. The basic mechanism layer provides a layer of abstraction between the OSS applications, OSS framework and telco infrastructure. Hence, OSS applications and OSS framework services invoke infrastructure functions via the basic mechanisms layer. The basic mechanism also contains capabilities to abstract access to underlying OSS/BSS and telco infrastructure systems. This layer is implementable as a transport mechanism, using various technologies such as the web services [7] Enterprise Service Bus (ESB) or CORBA's [83] Object Request Broker (ORB).

Accessed and used across all layers in the TNA, is the policy/security functions and the shared information and data repositories. The policy/security functions ensure each layer and its contents operates in accordance to rules and regulations defined by the telco. The shared information and data repositories administer diverse information and data used by layers' services and functions. Hence, the appropriate information and data is securely and easily accessible by the authorised services and functions throughout any layer of the TNA.

eTOM business processes, SID information processing objects and the corresponding TNA components are implementable using various technologies. Currently, implementations use web service [7] technologies. We discuss web services in *Chapter 7*.

6.3 Reusable Concepts

TMN, TOM and eTOM define requirements that a telco OSS/BSS must satisfy. These requirements are structured using a *business model*, containing interacting business entities. These interactions represent relationships between the business entities. All relationships are standardised using reference points. Consequently, TOM and eTOM decompose their business entities and reference points into standardised interacting business processes, that support their business relationships.

Generic and reusable *services* are defined by the management architectures, thought in different forms. TMN defines management functions that are integrated to form management applications. These functions are complex with protocols supporting their communication. Also, functions and protocols are network dependent. TOM and eTOM use abstract business processes as services that manage telco infrastructure, business partnerships and ultimately business objectives. Business processes interact using information flows that are network and technology neutral. Thus, business processes as services hide the underlying complexity of the OSS/BSS implementation and the telco infrastructure.

Layers are used as a modeling tool to simplify the complexity of the management architectures. In TMN, TOM and eTOM, layers are used to *separate* the various services according to the functions they provide. As a result, a hierarchy of service abstracts is created. For example, TMN layers management functions with business and service functions forming the higher layers and network and element functions forming the lower layers. Also, TOM layers its business processes according to their capabilities. For instance, business processes involving customers form the topmost layer, while network-oriented business processes form lower layers. eTOM uses a similar layering scheme as TOM. However, the eTOM layers its partner-oriented business processes at the bottom, since partner interactions may be required to fulfil higher business, service or network-oriented business process requests.

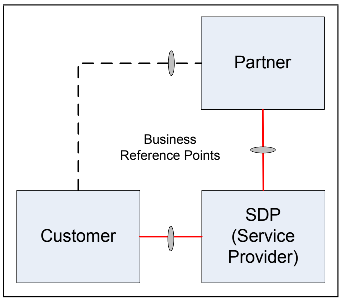
Each of the management architecture's layers contain varying functions. In these architectures higher layer functions abstract access to lower layer functions. For example, TMN's service management layer contains service management functions that abstract access to lower network management functions and their information. Also, when an eTOM customer layer business process is executed its output may invoke lower service layer business processes. Some of the architectures layers expose access to their functions. This is seen in TOM and eTOM layers that expose business processes to customers, with eTOM also exposing business processes to diverse partners.

Interaction between management functions occur within and across layers. In TMN communication between functions are formalised using *reference points*. These reference points promote standardisation since they are decomposed into complex protocol specifications. Both TOM and eTOM specify business reference points between business entities contained in their business model. These reference points are decomposed into interactions between various business processes. By applying the SID to the eTOM business processes, information required for these interactions are formalises as APIs. These APIs also contribute to the definition of the TNA.

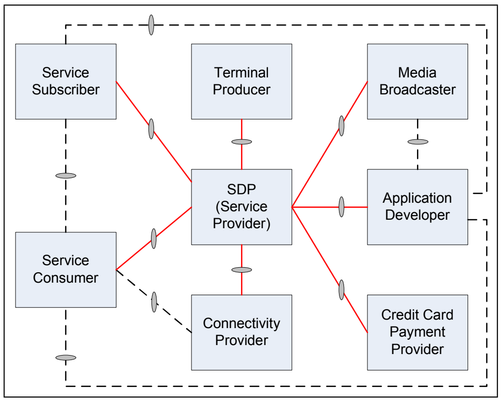
TMN, TOM and eTOM use *domains* in their architectures to illustrate the distributed nature of the telco OSS/BSS. For instance, the business models and architectures show the telco interacting with various external partners, via its OSS/BSS. These partners may operate as external telcos or IT-using enterprises. However, TOM and eTOM show the internal distribution of business processes across the telco OSS/BSS by using process flow domains. These domains represent the grouping of business processes across layers that aim to fulfil specific business objectives. The process flow domains also indicate business process implementations may be distributed across the telco OSS/BSS.

The telco management standards abstract the complexity of their implementations and distribution to various degrees. TMN uses complex protocols to abstract implementation of management functions. However, these protocols do not abstract distribution of the functions. The eTOM's corresponding TNA promotes the use of *middleware* to abstract the various underlying complexities of implementing the distributed telco OSS/BSS. In addition, the TNA uses middleware to abstract integration of various technology specific OSS/BSS vendor products. This middleware is the product of the both TNA's OSS framework layers and the basic mechanism layer.

As middleware, the OSS framework layers provide generic and reusable services that enable business processes and OSS applications to locate each other and invoke their functions. Also, the OSS framework layers contribute middleware services that provide OSS-specific functionality to business processes and OSS applications. Some of these middleware services provide functionality such as to monitor underlying elements and log interactions between software components. As middleware, the basic mechanism provides the reliable, secure and guaranteed transportation of requests between business processes and OSS applications. Also, the basic mechanism integrates with telco infrastructure systems, therefore enabling OSS applications to invoke telco infrastructure management functions. Therefore, the TNA middleware abstracts the distribution and technologies used to implement the OSS/BSS business processes, applications and telco infrastructure. As a result, the middleware provides the needed capabilities to support the business, service and customer-oriented perspectives on the telco and its OSS/BSS, by abstracting various underlying complexities.



(a) Generic SDP Business Model derived from TMN, TOM and eTOM Business Models



(b) Example of a Specific SDP Business Model

Figure 6.5: SDP Business Models

6.4 Contribution to the SDP from TMN, TOM and eTOM

We reuse the generic TMN, TOM and eTOM (including TNA) concepts to define a SDP business model and management architecture. The SDP business model is shown in *Figure 6.5(a)*. Also, the SDP business model is derived from the TMN, TOM and eTOM business models. The SDP business model is highly abstract and extendable to accommodate TMN, TOM and eTOM business models.

The SDP business model defines three business entities that are crucial to the telco business and network operation. These entities are customers, partners and the SDP itself. The SDP represents the telco service platform that exposes various management services to customers and partners. Thus, the SDP business entity is similar to the service provider found in the TOM and eTOM business models.

Customers represent business entities that use the SDP for access to telco, IT and Internet-based services. Some of these services provide customer and service management functionality, such as customer registration, service subscription and service configuration. The customer entity may be decomposed into more specific business entities. For example, customers may include other service provider enterprises, who subscribe and resell services to individuals or other enterprises. The customer business entity is similar to customers found in the TOM and eTOM business models.

SDP partners are diverse, ranging from application providers, content providers, connectivity providers and even terminal manufacturers. Partners use the SDP and its services to develop and manage customer services. In addition, the partners use SDP services to manage their business relationship with the telco. For example, content providers use SDP services to register their content with the SDP. In addition, application provides use SDP services to charge customers for using their customer services. Thus, the partner business entity represents an abstraction of numerous business entities found in both TOM and eTOM business models. For example, partners include suppliers, intermediaries and complimentary providers.

Like TOM and eTOM business models, the SDP business model structures interactions between business entities using business reference points. However, for the SDP business model we use more abstract representations for reference points, called business relationship points. These business relationship points describe the allowed behaviour between business entities. For example, business relationship points are defined as formal contracts between SDP and telco partners. These contracts specify details and the terms and conditions of the interactions. Like the business entities, these business relationship points are decomposable. An example of a decomposed business model is shown in *Figure 6.5(b)*. In the figure we decompose the customer and partner business entities. Decomposed customers include a service subscriber and service consumer. Decomposed partners include terminal manufacturers, media broadcasters, application developers, a credit card payment provider and a connectivity provider. The figure also shows numerous business relationship points between the SDP and the decomposed customers and partners. However, unlike TOM and eTOM we only consider business relationship points involving the SDP (service provider).

Based on the generic SDP business model, we define a SDP management architecture shown in *Figure 6.6*. The SDP management architecture is derived from the TMN, TOM and eTOM architectures. The SDP management architecture structures the SDP within an environment containing various management entities. By invoking APIs, the environment's entities are able to use management capabilities offered by the SDP.

The SDP management architecture decomposes business entities and their business relationship points into business processes, management services, management functions and APIs. In the management architecture, layers are used to group the various business processes, management services and management functions. The business process layer contains generic business processes, used by the telco to manage business, service, customer, partner and network operations. These processes are also accessed and invoked by customers and partners that are involved in business relationships with the telco. The management service layer contains abstract management functionality that is used by business processes to fulfil telco, customer or partner requests. These applications represent the full capabilities of the telco OSS/BSS. The management function layer contains complex management functions that enable the underlying telco infrastructure systems to be used in satisfying telco business objectives. The management architecture also defines a converged

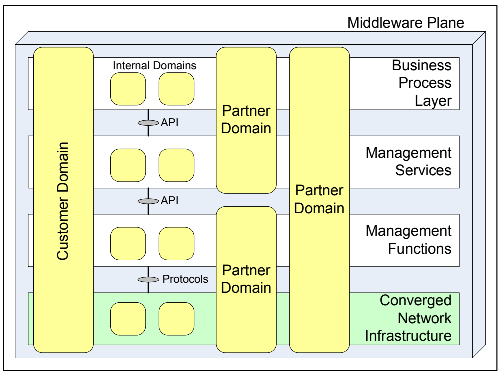


Figure 6.6: SDP and its Environment derived from TMN, TOM and eTOM Architectures

infrastructure layer that represents the management resources and capabilities offered by the underlying networks. This layer includes telco, IT and Internet-based networks. Interactions and communication between layers are supported using their contents APIs.

The SDP management architecture uses domains to distribute management functions across telco infrastructure, customer terminals and partner infrastructure. These domains relate to the SDP business model's entities, that is, customers and providers. The TMN also use domains to distribute its architecture, while TOM and eTOM use domains to group their business processes into end-to-end process flows. The SDP management architecture shown in *Figure 6.6* illustrates examples of customer domains and multiple partner domains. The customer domain expands across all layers, since the customer terminal may contain management services or functions that invoke functionality in the corresponding SDP management architecture layers. The partner domains vary depending on the type of partner and functionality required by the partner. For example, an application provider may require access to service subscription business processes. Also, the application provider may provide billing information to telco management services. As a result, the application provider domain overlaps onto the business process and management service layers. Other types of partners may require access to lower layer management capabilities. For example, connectivity providers may require access to telco networks, but also provide routing information to management functions. Hence, the connectivity provider domain overlaps onto the management function and network layers. In the SDP management architecture we also show partners that use all SDP management capabilities. As a result, some partner domains extend across all architectural layers.

Besides partner domains, the SDP management architecture illustrates internal domains located within most layers. Like TOM and eTOM process flows, the SDP management architecture uses internal domains to represent groupings of various SDP management capabilities. Internal domains group specific SDP management capabilities such that they are used in completing specific business, service, customer or network management activities.

For example, business processes that aid service billing may be grouped into an internal service billing domain. The corresponding management services and functions that aid these business processes are also grouped into the internal service billing domain. Some of the business processes, management services and management functions may also be distributed across external customer or partner domains. Hence, internal domains may span across multiple layers and external domains of the SDP architecture.

Borrowing concepts from eTOM and TNA, we use a middleware plane to support the distributed nature of the SDP management architecture. The middleware plane encompasses all SDP capabilities, layers and domains. The middleware plane hides the distribution of management functions contained in all layers and domains. As a result, business processes, applications, services and the telco infrastructure systems interact independently of their physical location. In addition, the middleware plane provides implementation independence, by abstracting the technologies used to realise layers' capabilities. Hence, the middleware plane hides technology specifics of business processes, applications, services and the telco infrastructure systems, but helps in exposing their technology neutral APIs.

Like the TNA framework layers, the middleware plane provides its own services. These services provide management functions that are used across all of the SDP's layers. For example, these services enable SDP management capabilities to operate across a distributed environment, by abstracting computing platforms and transport networks that constitute the telco infrastructure and OSS/BSS. Also, these services may abstract access to legacy telco OSS/BSS system functions. The middleware services offer access to their functions by exposing technology neutral interfaces (APIs).

6.4.1 Defining a Complete and Managed SDP Architecture

In *Figure 6.7* we illustrate a complete SDP architecture that incorporates the SDP management architecture layers. The figure also shows a managed SDP contained within its environment. The SDP uses its various service interfaces to interact with its environment.

The SDP architecture in *Figure 6.7* extends management layers to interwork with non-management layers. These non-management layers are the exposed services layer, application service layer and network function layer. The exposed services layer houses SDP services that are used by external telco partners to deliver content to customers. Also, these services are used by telco partners in application development, execution and delivery. The application service layer contains application services that support the exposed services. Application services provide exposed services with simplified access to underlying telco network resources and capabilities. The network function layer houses network functions

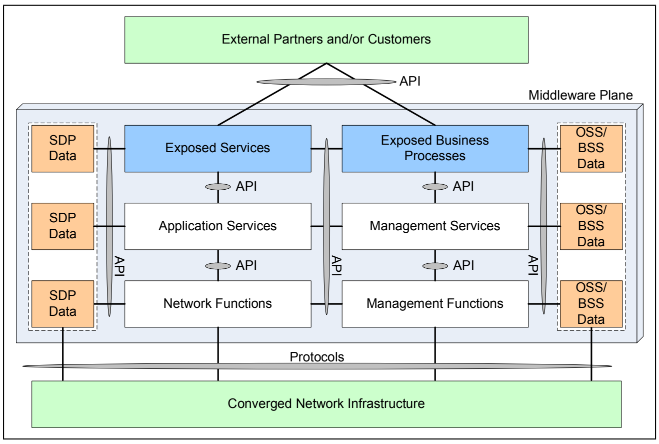


Figure 6.7: A Managed SDP and its Environment

that support application services. The network functions provide application services with access to the rich functions offered by the telco's physical network elements.

Vertical communication between the exposed services, application services and network functions occur via their APIs. Additional service APIs are also used to enable horizontal communication with the management layers. We name these APIs service management APIs. Exposed services and business process layers communicate via their service management APIs. For instance, this communication ensures usage of exposed services trigger various billing, monitoring or security business processes. Application service and management services layers also communicate via their service management APIs. For example, management services may detect and halt application services from accessing unavailable or faulty network functions. Network function and management function layers communicate via their service management APIs. As an example, network functions may request management functions to configure transport networks so as to deliver content with the appropriate quality of service.

The SDP architecture extends all layers to include data services that are accessible via their APIs. Two types of data services are defined: OSS/BSS data services and SDP data services. OSS/BSS data services manage abundant OSS/BSS data required in the execution of business processes, management services and management functions. The SDP data services are used across all non-management layers and provides data used in the execution of expose services, application services and network functions.

Like the SDP management architecture, the complete SDP architecture uses a middleware plane to abstract service implementations and their distribution. Thus, the SDP architecture does not indicate the domain distribution of layers. Also, *Figure 6.7* shows the middleware plane abstracting the underlying technologies used to implement each layer, its services and their APIs.

6.5 Evaluation of SDP Business Model and Architecture

By extracting, extending and applying the generic TMN, TOM and eTOM concepts to the SDP we have defined a management-oriented viewpoint. This viewpoint defines various management abstractions that are structured in the SDP business model and management architecture. The management abstractions contribute to the definition of the SDP framework. We evaluate the SDP business model and management architecture against the generic concepts extracted from TMN, TOM and eTOM. In addition, the evaluation provides answers to the questions posed in *Chapter 1 Section 1.3*. We also present the comparison between TMN, TOM, eTOM and the SDP management architecture in *Table 6.1*.

The SDP business model illustrates management *requirements* that the SDP must satisfy. These include supporting business relationships with multiple diverse partners and customers. Hence, the business model defines business requirements, describes telco/SDP business objectives and enforces the telco business strategy within the converged telecom, IT and Internet market. In addition, the business model enables the SDP to focus on satisfying customers and partners by using its services. The business model formalises business relationships between SDP, telco, customers and partners using business relationship points. These business relationship points standardise interactions between various business entities. Both business entities and business relationship points are extendable.

The SDP uses its vast repository of *services* to implement business relationship points. The collection of services represent the SDP's business processes, management applications and management services. These services represent the capabilities offered by the SDP to satisfy business requirements, objectives and strategy. Various services are offered to customers and partners. Also, services are used internally to simplify access to management functions offered by telco infrastructure. Services expose their functionality via their APIs. Service APIs are rich since they express management functionality offered by the SDP. In addition, the APIs enable the vast amounts of telco management information to be obtained, simplified, structured and used. Therefore, SDP services and their APIs further contribute to the standardisation of business entity interactions since they implement business relationship points.

Standard-based service APIs promote SDP *standardisation*. Interoperability between SDP, partner and customer infrastructures is made easier with standard-based SDP service APIs. Also, customers and partners may easily move between SDP implementations if it exposes standard-based service APIs. Service APIs simplify access and usage of the underlying network resources, since resources communicate using complex standard-based protocols.

	TMN	TOM/eTOM	SDP	Description
Business Model	Limited to telcos only	Business Model with reference points between various customers and partners.		Summarises requirements. Formalises relationships. Also, promotes standardisation when implementing reference points.
Layers	Business Layer	TOM/eTOM-based Business Processes	Business Process Layer	Contains business processes that are accessible by customers and partners.
	Service Layer	SID and TNA derived OSS Applications	Management Service Layer	Contains management services that provide access to OSS/BSS functions.
	Network Layer		Management Function Layer	Provides the common standard-based interface to the technology specific OSS/BSS capabilities.
	Element Layer			
	Network Element Layer			
	IN/PSTN	Network Resources	Network Resource Layer	Represents the complex underlying converged systems that must be managed.
Domains	Limited telco domains	Internal and external domains that include diverse partners and customers		Distributes layers internally and externally across telco SDP and enterprise infrastructure.
Middleware	Complex Protocols	TNA Framework and Basic Mechanism	Middleware Plane and Services	Provides a mechanism and services that abstracts complexities of implementation, distribution and technology.

Table 6.1: Comparison of Management standards and the SDP Management Architecture

Hence, standard-based service APIs provide consistent access to lower network management capabilities, independently of network technologies.

The SDP management architecture shown in *Figure 6.6* uses layer, domain and plane *design patterns* to structure SDP services and their APIs. This architecture expresses the management functionality of the SDP. As a result, it extends the proposed SDP architecture in *Figure 1.3* by using SDP services and their APIs to promote standard-based interactions with telco OSS/BSS and partner SCE/SME.

In the SDP management architecture layers are used to group SDP services into various management categories. These categories are business processes, management services and management functions. As a result, layers *separate* SDP management capabilities into a service hierarchy. Layers at the top of this hierarchy simplify access to lower layers. Also, higher layers simplify information obtained from lower layers and network systems. Hence, customers and partner access SDP capabilities using services found at the topmost business process layer. Layers at the bottom of the hierarchy are used to administer telco infrastructure systems. Each of the architectures' layers are highly generic and may be further decomposed into more specific layers. For example, as in TOM and eTOM the business process layer may constitute specific customer, service and network oriented business process layers. These new layers separate and group business processes according to the customer, service or network management capabilities they provide.

Domains *distribute* SDP services and layers across various telco and IT-based infrastructures. Some external infrastructures may belong to partners and customers. For example, partners include IT-using enterprises, such as application developers. Also, customer infrastructure includes the diverse customer terminals or IT-based enterprises that operate on behalf of customers.

The middleware plane is used throughout the SDP management architecture to ensure distribution and implementation complexities are *hidden* from customers, partners and SDP services across all layers and domains. Also, the middleware plane abstracts interactions between customers, partner and SDP services by hiding computing platforms and transport networks used to deliver communications. The middleware plane provides a wealth of reusable services that extend the management functionality offered by the SDP. For example, middleware services may be used to locate and register for SDP services. Also, some middleware services may be used to enforce usage policies for partners accessing SDP service. Middleware services are accessed across all layers and domains by using their technology neutral APIs.

The additional SDP architecture shown in *Figure 6.7* consolidates SDP management services with other types of SDP services. For example, a single layer may be used to house exposed services and business processes, since they are invoked by external partner applications. Also a layer containing both application services and management services represents a *point of integration* with converged network resources and capabilities. Combining network functions and management functions into a single layer provides a consistent interface to access network functions and manage physical network elements. The SDP architecture also identifies APIs between management SDP services and non-management SDP services. These APIs promote standard-based communication between the SDP services. In addition, the SDP architecture identifies management and non-management data that is access via SDP data services contained in the various layers. These SDP data services also provide APIs to their functionality. These data service APIs promote standard-based access to OSS/BSS and SDP data.

6.6 Summary

In this chapter we presented contributions of telco OSS/BSS standards to the SDP and its framework. We reused generic concepts from the TMN, TOM and eTOM (TNA) standards. By reusing these concepts we defined a SDP business model and management architecture. The SDP business model contains abstract business entities involved in business relationships. We formalised the business relationships using generalised reference points called business relationship points. Similar to reference points, business relationship points promote SDP standardisation. The business relationship points are realised using SDP services, that expose access to their functions using implementation and distribution independent APIs. We presented various types of SDP services that aid telco business and operations management. The SDP management architecture uses horizontal layers and vertical domains to distribute SDP services across various internal and external functional areas. These divisions extend into external infrastructures. We used a middleware plane to hide distribution complexities of SDP services, when accessed via their APIs from various external infrastructures. We defined the middleware plane to provide a variety of management services that are accessible across all layers and domains. We consolidated the SDP management architecture into a more complete SDP architecture. The SDP architecture uses common layers to group both management services and non-management services belonging to the SDP. The architecture also identified SDP data services that must be added to layers, so as to abstract access to both OSS/BSS and SDP specific data. We also used a middleware plane to abstract various distribution complexities in the SDP architecture. Therefore, by reusing generic TMN, TOM and eTOM concepts we presented technology neutral abstractions and architectures that contribute to the SDP framework.

Chapter 7

Perspective on the SDP from an Enterprise Standard: SOA

The public telco network provides voice and data services to its customers by using heterogenous technologies. These technologies include voice service platforms, circuit-mode transport networks and specific customer terminals. However, the telco also operates as an enterprise. Similar to other private enterprises, the telco uses *IT-based* solutions to manage its operations and business.

Traditional private enterprise networks support networked IT solutions to automate business processes, such as Customer Relation Management (CRM) and Enterprise Resource Planning (ERP) solutions. IT is also used to extend the business to customers and business partners. For example, customer services are delivered over the Internet and communication with partner IT solutions occur using Enterprise Application Integration (EAI) middleware. Many of the IT solutions are either standards-based or proprietary.

Both telco and enterprise networks operate in distinct markets, are regulated to differing extents and provide their own specialised services to customers and partners. In addition, each network uses specific infrastructure to develop, deploy, deliver and manage its services. Currently, both telco and enterprise networks, their infrastructures, customer requirements, business markets and regulations are *converging*.

To manage and benefit from convergence the enterprise uses the *Service Oriented Architecture (SOA)*, while the telco aims to use the SDP. The SOA simplifies the integration of new and legacy IT solutions, so as to satisfy new business process requirements [113]. Also, the SOA is based on the web service [7] technology standards. Though technology-based, most of the SOA concepts are technology independent. In addition, these concepts are applicable to the SDP, since they both aim to manage convergence. Hence, we extract and reuse the generic technology neutral SOA concepts to define an IT enterprise perspective on the SDP.

7.1 Requirements

As defined by [7], the SOA represents a distributed system architecture that describes the structure of an enterprise's converged infrastructure. The SOA illustrates this structure as a wealth of interacting services that have the following properties. SOA services:

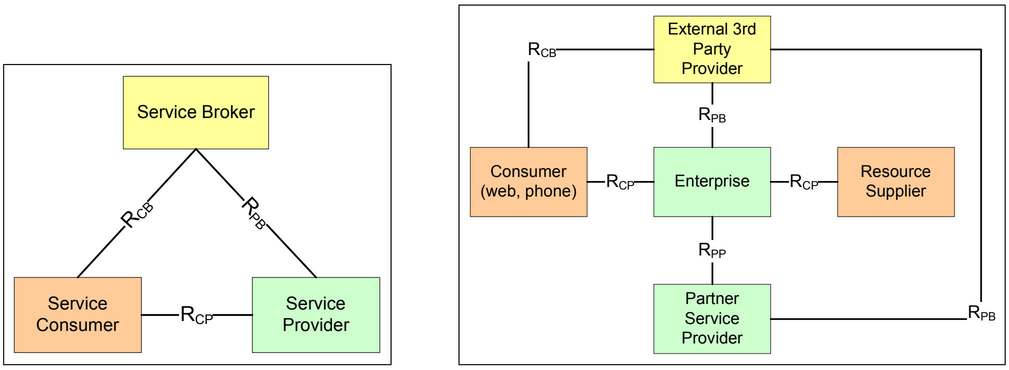
- abstract functions provided by infrastructure resources and capabilities in a technology neutral manner;
- communicate via technology-neutral message flows;
- may be distributed across diverse networks; and
- describe and expose their capabilities in a technology-neutral manner.

Hence, these service properties must be satisfied for a SOA to be defined by an enterprise.

Currently, the SOA is the web services [7] architecture. As a result, the SOA's services are defined as web services. These web services abstract access to enterprise infrastructure functions, such as data stores, communication buses and legacy applications. Also, web services define their functions in a technology neutral interface, using the XML-based WSDL. Communication between web services occurs using messages conveyed in the technology neutral SOAP protocol. Also, these web services may reside internally or externally across the enterprise infrastructure.

The SOA supports the basic business model shown in *Figure 7.1(a)*. The business model is derived from [7]. In the business model, three business entities are defined: a service provider, service consumer and service broker. The service provider represents an enterprise that creates web services to abstract its infrastructure's hardware and software capabilities. The service broker is used by the service provider to register its web services location and interfaces. The broker is used by the service consumer to find the appropriate web service and interface. Once found, the service consumer invokes the service provider's web service interface. The service broker and consumer may be external or internal to the enterprise. We represent interactions between business entities as business relationship points. These business relationship points formalise business interactions between the business entities.

The business model shown in *Figure 7.1(a)* is highly abstract. Most enterprises incorporate this SOA business model by identifying service providers, consumers and brokers in their own business model. Once identified, the enterprise can define the interactions or business relationship points between the business entities. The business model describes only the business objectives of the enterprise. To realise the business model, the enterprise must



(a) Service-Oriented Business Model (b) Example of a Service-Oriented Reference Model

Figure 7.1: SOA Business and Reference Models

implement the various services. To initiate the implementation, the business model is decomposed into a reference model. An example reference model is shown in *Figure 7.1(b)*.

The example reference model is defined by decomposing the business model's business entities and business relationship points to reflect additional details on the enterprise, its customers and partners. Business entities are decomposed into business objects. Business objects represent components of a business entity that interact with other objects in the business model. In the reference model the service provider is decomposed into the enterprise and partner service provider. The service broker business entity is decomposed into the external 3$^{\text{rd}}$ party provider business object. Also, the service consumer business entity is decomposed into the consumer and resource supplier business objects. The correspond business relationship points between the business entities are also decomposed into reference points. Reference points promote standardised communication between business objects, by formalising their interactions. In the reference model, their are one-to-one mappings of business relationship points to reference points. However, we define the additional R_{PP} reference point to formalise interactions between two service providers business objects, that is, the enterprise and partner service provider.

With business objects and reference points defined, the reference model is further decomposed. Business objects are decomposed into web services that are offered to other business objects. Reference points are decomposed into the corresponding web service interfaces. The interfaces describe functions provided by the web services. Hence, execution of various web services via their interfaces contribute to the fulfillment of the business objectives defined by the business model.

Various benefits are gained by an enterprise implementing a SOA. For instance, the use of web services enables faster application development [114], since web services are easily orchestrated via their interfaces into applications. Also, the SOA and its web services decrease infrastructure complexity, enable reuse of existing infrastructure functions and support integration with external infrastructures [115]. Hence, the SOA enables enterprise agility, by restructuring legacy enterprise infrastructure into a communicating network of web services [116].

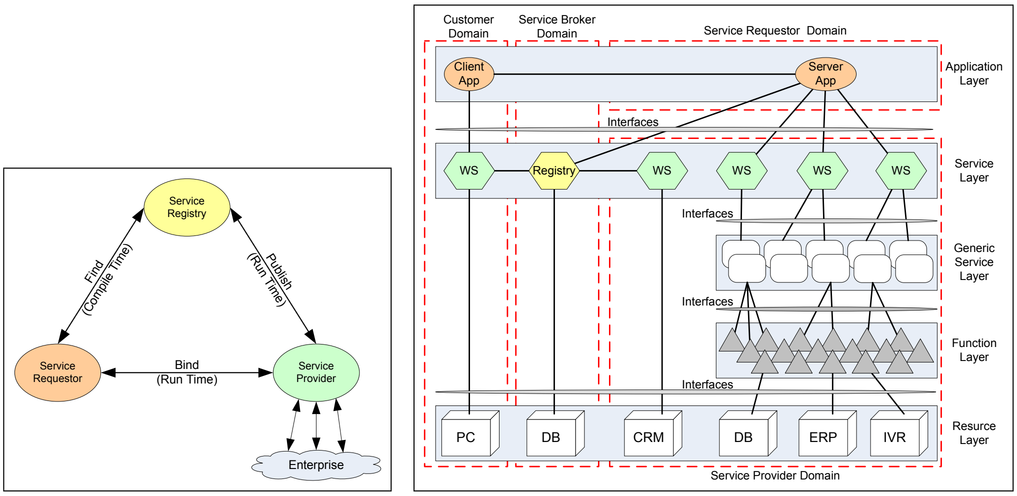
7.2 Architecture

For the SOA, two forms of architectures exist. First is the standard-based web services SOA and second the enterprise SOA.

7.2.1 Web Services SOA

The SOA is illustrated in *Figure 7.2(a)*. This architecture relates to the SOA business model described previously. Web services are created by a service provider, that uses a *Universal Description Discovery Integration (UDDI)* [15] service registry to *publish* a description of its web service interface. The interface is described using WSDL. When a service requestor requires a web service's functionality, the requestor uses the UDDI registry to *find* the appropriate interface description. Once obtained, the requestor will use information contained in the description to *bind* to the web service and use its functionality. Once bound to the web service, the service requestor uses SOAP messages to invoke the web service functions, as defined by its interface description. All SOAP messages are transported using HTTP. Based on the architecture, the web services interface implements the SOA reference model's reference points.

We provide additional structure and detail on the SOA shown in *Figure 7.2(a)*. This extended architecture is shown in *Figure 7.2(b)*. In the figure, we use layers to separate the components of the SOA. The top most layer contains applications that use web services to invoke enterprise infrastructure functions. The service layer contains the web service registry and web services that are accessible to applications. The generic service layer contains enterprise infrastructure capabilities that are invoked by web services. The function layer contains abstractions of infrastructure resources, such as billing systems, databases or legacy applications. These functions are used by the generic services to fulfil web service requests. The lowest layer contains the physical enterprise infrastructure resources and capabilities.



(a) Web Service SOA (b) Extended Web Service SOA

Figure 7.2: Web Service Standard-based SOAs

In *Figure 7.2(b)*, we also show domains that applications and web services must operate across. These domains correspond to the business entities that are defined in the SOA business model. For example, applications operate in the service requestor domain, but access web services located in the service provider domain. Also, the architecture includes a customer domain with customer applications, who invoke server applications located in the service requestor domain. The use of domains in the architecture reinforces the distributed nature of the SOA. Hence, the SOA must implement mechanisms to manage the various complexities that are introduced by distribution.

We define the extended SOA to support *horizontal* communication between entities within the same layer but across different domains. For instance, client applications may invoke a server application within the application layer, but across customer and service requestor domains. Also, we define *vertical* communication between entities located across different layers and domains, such as service requestor applications invoking service provider web services. Most horizontal and vertical communication occurs via interfaces that are internal to a layer or externally exposed by a layer. For example, the service layer exposes web service interfaces to applications. Also, within the service layer the registry service communicates with web services via their interfaces. Some lower layer interfaces are highly complex and use protocols or technology specific mechanisms to communicate. Examples of complex interfaces include those exposed by the underlying technology specific enterprise infrastructure.

The architecture in *Figure 7.2(b)* illustrates two web service deployments across the enterprise. First, web services are used to directly interface with the physical enterprise resources. As a result, these web services and their interfaces abstract complex interactions with the enterprise capabilities. Second, web services are used to abstract legacy and new software-based enterprise applications and services contained in the generic service layer. These applications and services are also complex to use, since their interfaces are technology diverse. As a result, these applications and service interfaces require simplification to access and use their functions in a technology consistent manner. Hence, web services abstract the complexity of these existing enterprise applications and services using its interface. Examples of both deployments are seen in telecom standards, such as Parlay X that is described in *Chapter 5*.

In addition to telecom standards, vendors employ both deployment options for their SDP products. That is, vendors provide web service interfaces to their SDP solutions, such that web services hide the underlying complexity of the software components and physical hardware. As a result, there is a growth in popularity for web services to be used in telco SDPs. This popularity has also spread to the development of other telco enterprise solutions, such as web service interfaces for OSS/BSS systems.

7.2.2 Enterprise SOA

Enterprises require agility to benefit from convergence opportunities. However, legacy IT solutions, fused with specific technologies, hinder agility. The legacy solutions represent a collection of automated business processes. To simplify these solutions and gain agility, the SOA is used. The objective of using the SOA in an enterprise, is to support an enterprise-wide transformation, where independence of new and legacy technologies are gained from implemented business processes [113]. This is achieved by wrapping legacy solution functions into web services. These web services enable a substantial amount of enterprise infrastructure investment to be reused.

The SOA satisfies most business process management requirements [116]. For example, it abstracts existing management system functions into reusable web services. Some web services may be created by the enterprise or bought from vendors. With web services defined, the enterprise may quickly create new business processes and implemented them as applications. Hence, business processes are implemented as a collection of web services that execute in a predefined order. Existing technologies, such as the *Business Process Execution Language (BPEL)* [117], may be used to create these business process applications. The SOA and its web services may also be used to improve or streamline existing business processes.

Both web service and enterprise SOAs are distributed within and across diverse infrastructure. However, both SOAs enforce distribution independence. The web services SOA uses

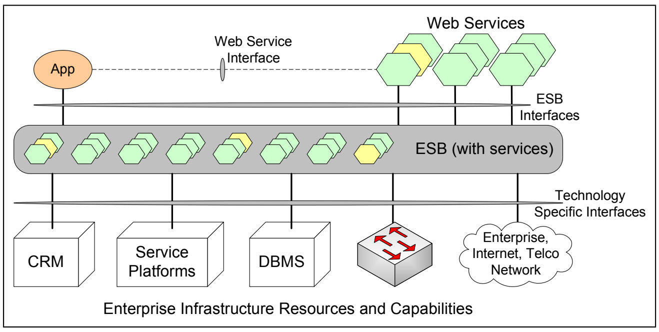


Figure 7.3: Enterprise SOA Representation

complex mechanisms to provide limited distribution independence. These mechanisms include UDDI registries and web-based protocols. However, the enterprise SOA provides a dedicated mechanism to support distribution and technology independence. This mechanism is generalised as the *Enterprise Service Bus (ESB)*. ESBs support web services by providing a range of functionality to enhance web service deployments and operations. For instance, [118] defines ESBs to provide:

- a scalable, high performance, robust and secure IT-based platform;
- standards-based communication mechanisms that connects applications, web service and infrastructure functions;
- real-time and reliable connectivity between enterprise infrastructure, the Internet, telco networks, applications and web services;
- data transformation, such that communication between different application and web service implementations are achieved; and
- support portability of applications and web services across various platforms and ESBs.

In addition to the above properties, ESBs provide services that are used to support applications and web services. These ESB services may provide functionality to resemble a UDDI registry, manage underlying communication mechanisms and configure physical enterprise equipment. ESB services may be implemented as web services that offer access to their capabilities via their interfaces.

We illustrate the enterprise SOA in *Figure 7.3*. As shown in the architecture, the ESB abstracts the various layers and domains of the extended web services SOA. Thus, the ESB enables the enterprise to perceive a collection of web services, independently of their location, implementation and infrastructure technologies.

7.3 Reusable Concepts

The SOA business model uses basic business entities to manage and benefit from convergence. In addition, the business model *formalises* interactions between business entities using business relationship points. These points define business contracts between the business entities. The SOA business model is generic and *extendable* to include various enterprise customers and partners. Also, the SOA business model structures and visualises the enterprise's business objectives. These objectives are satisfied by implementing the SOA.

The SOA reference model that is derived from the business model, provides an important milestone in realising a SOA implementation. The decomposition of the business entities and business relationship points illustrates *separation and abstraction* of enterprise infrastructure capabilities. This is achieved by decomposing business entities into objects that provide a variety of services. Also, the business relationship points are decomposed into reference points that formalise services' interfaces. These interfaces structure and formalise communication between the various business objects. Hence, the reference model ensures business objectives are satisfied by defining the required service interfaces and promoting their *standardisation*.

Both web service and enterprise SOAs are derived from the SOA reference model and provide a *service-oriented perspective* on the enterprise. Each SOA uses web services to abstract capabilities from enterprise infrastructure. These web services enable easier access to infrastructure capabilities. Also, web services may be used to abstract existing enterprise applications and services. However, these SOAs are tightly coupled with web service technologies, specifically WSDL, SOAP and HTTP. We extract the generic concepts from the web service and enterprise SOAs without being bound to the particular technologies. As a result, we define the *Generic SOA (GSOA)*. A GSOA representation is illustrated in *Figure 7.4*.

The GSOA is a distributed system architecture containing serving entities, called services. These services offer functionality defined as technology independent interfaces. Applications access service functionality by orchestrating service interfaces, independently of location and implementation. Both applications and services are supported by a distribution mechanism, called the distribution plane. In *Figure 7.4*, the GSOA services abstract access and usage of the underlying enterprise infrastructure resources and capabilities. Also, services hide details of their implementations using their implementation neutral interfaces. Hence, service implementations may be accessed consistently via their interfaces, by various application implementations. The figure also shows the GSOA distribution plane hiding technology details of the underlying network and systems from the various applications and

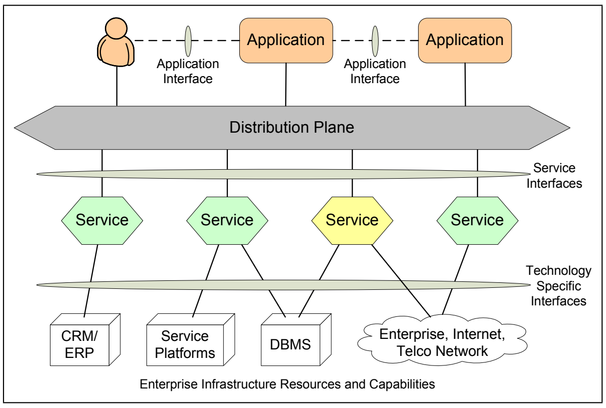


Figure 7.4: Generic SOA Representation

services. The distribution plane also provides its own services that may be used by applications and services to fulfil their functions.

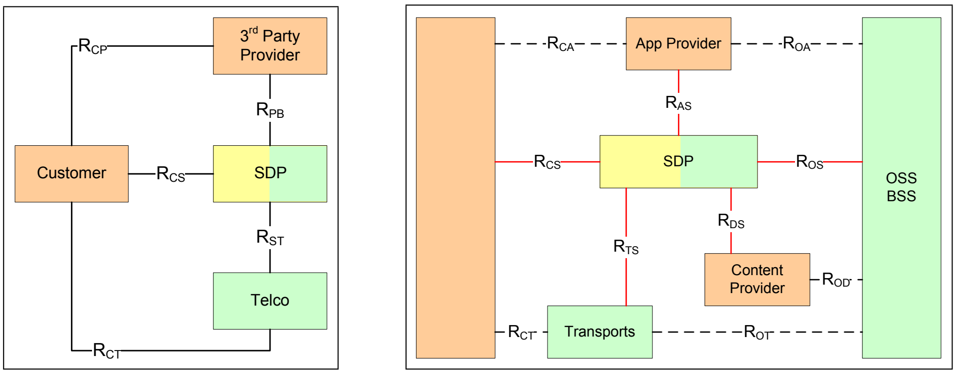
Based on *Figure 7.4*, the GSOA services are used for abstracting access to new and legacy databases, CRM and ERP solutions. This abstraction is also encountered when implementing the SDP in the presence of legacy OSS/BSS technologies and solutions. Thus, a GSOA is suitable for abstracting the telco OSS/BSS because of the large amount of services it provides. Also, GSOA services may be used to abstract access to new and legacy telco service platforms, such as the IN. These services support application development, such that new customer services may be created. Thus, a GSOA is also suitable for abstracting the telco service platforms because of the large amount of services that application developers require.

Since it is technology neutral, The GSOA and its concepts are reusable by any enterprise and implemented using different technologies. For example, the web service and enterprise SOAs are specialised forms of the GSOA. As a result, the GSOA may be used to structure a SDP architecture.

7.4 Contribution to the SDP from the SOA

We reuse the generic SOA concepts to define a SDP business model. In addition, we decompose the business model to define a SDP reference model. We also use the GSOA to structure a SDP architecture. The SDP business model is shown in *Figure 7.5(a)*.

The SDP business model is derived from the SOA business model shown in *Figure 7.1(a)*. The SDP business model shows four main business entities. These entities are the SDP itself, external providers, the telco and customers. The SDP business entity performs two



(a) Generic SDP Business Model derived from SOA Business Model

(b) SDP Reference Model derived from SOA Reference Model

Figure 7.5: SDP Business and Reference Models

business roles. First, the SDP is a service provider that provides services to other business entities. Second, the SDP is a service broker that aids business entities to locate services offered across the business model. The 3$^{\text{rd}}$ party provider business entity performs the role of a service consumer that uses SDP services to provide additional functionality to both SDP and its customers. The telco business entity performs the role of a service provider since it represents the available telco infrastructure that the SDP services abstract. The customer performs the role of a service consumer that uses the SDP and its services to locate, consume and pay for customer services.

We decompose the SDP business model to illustrate the various business objects and reference points that promote the standardisation of the SDP. The SDP reference model is shown in *Figure 7.5(b)* and derived from both SDP business model and SOA reference model, shown in *Figure 7.1(b)*. In the reference model we decompose business entities into business objects and business relationship points into reference points. However, we focus only on reference points that include the SDP business objects.

In the SDP reference model, we decompose the 3$^{\text{rd}}$ party provider into application and content providers that use SDP services. Both application and content providers are essential to the SDP since they create and enhance customers services. The R_{PB} business relationship point is decomposed into the R_{AS} and R_{DS} reference points to formalise communication with application providers content provides respectively. The telco business entity is also decomposed into telco transport networks and the OSS/BSS. Transport networks are required by the SDP to deliver customer services. The OSS/BSS is critical to the SDP since it contains the telco business processes and manages the telco infrastructure. The reference model decomposes the R_{ST} business relationship point into the R_{TS} and R_{OS} reference

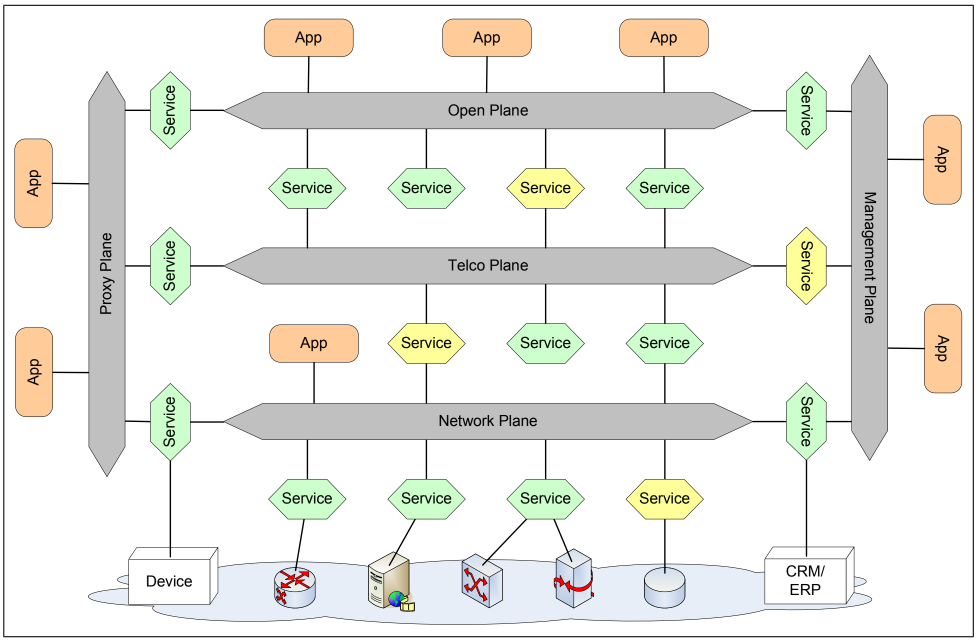


Figure 7.6: SDP and its Environment represented as Integrated GSOAs

points to formalise communication with the telco transport networks and OSS/BSS respectively. The customer and SDP business entities remain constant. However, these entities may be decomposed if the telco requires. The customer interacts with the SDP via the R_{CS} reference point. All reference points are implemented as SDP services with the appropriate interfaces. We structure the SDP services and their interfaces within a SDP architecture, by using the GSOA concept.

A SDP architecture based on the SDP business model and SDP reference model is shown in *Figure 7.6*. The architecture shows the SDP within its environment using GSOAs. The SDP architecture structures numerous SDP services and their interfaces. Different SDP services are identified based on the the telco infrastructure capabilities they abstract. Hence, SDP services are similar to SOA web services. As a result, the SDP architecture also groups services according to the different telco infrastructure capabilities they abstract. To structure and manage the collection of SDP services for different infrastructure capabilities, multiple GSOAs are used. Thus, the SDP architecture represents the integration of these GSOAs.

The GSOAs used for the SDP architecture contain services that wrap generic functionality from both legacy and new network resources, service platforms, customer devices and management systems of the telco infrastructure. Also, each of the GSOAs constituting the SDP implement one of its reference model's reference points. Also, business objects involved in a reference point relationship view the SDP as offering a single GSOA with numerous services, that hide underlying complexities.

In the SDP architecture, telco OSS/BSS functionality is wrapped into *management services*. These services implement telco operations and business processes, such as CRM, billing and network management. As a result, management services must use functionality provided by the network resources and service platforms. The products of integrating management services are *management applications*. Management applications and services are housed in a GSOA we call the *Management GSOA*. This GSOA implements the R_{OS} reference point.

Telco network resource functionality is wrapped into *network services*. Network service functionality includes manipulating traffic, configuring elements, processing protocols and managing network specific events. Also, network services may use management services offered by the management GSOA. The products of integrating network services are *network applications*. Network applications and services constitute a GSOA we call the *Network Resource GSOA*. This GSOA implements the R_{TS} reference point.

The SDP architecture shows service platform functionality being wrapped into *telco services*. Telco services simplify access to new and legacy platforms, such as IN. Telco services provide generic functionality to establish connections, coordinate sessions, manage state and manipulate databases. Telco services provide an abstracted interface to network services. Also, telco services may use management services offered by the management SOA. Telco services are contained in a GSOA we call the *Telco GSOA*. This GSOA implements the R_{AS} reference point.

Integrating both telco and management services, by application developers, into *converged applications* may be complex and time consuming. Also, developing new *telco applications* may be complex when reusing legacy platforms. In addition, content providers require simple functionality to register their content with the SDP. Hence, simpler services are needed to support application developers and content providers. To satisfy these requirements *converged services* are provided. Converged service functionality includes subscribing for services, registering content, making any call, streaming content and charging an account. Telco applications, converged applications and converged services constitute a GSOA we call the *Open GSOA*. This GSOA extends the R_{AS} reference point and enables the R_{DS} reference point.

The SDP architecture enables customers to access the appropriate GSOA and applications. Though customers are external to the telco, their devices may contain customer applications that aid in providing access to the SDP. Customer applications may use services offered by the device to communicate with SDP services contained across the various GSOAs. These are *customer services* that provide functionality to authenticate the customer, establish connections and process signalling protocols. Customer applications and services are housed in a GSOA we call the *Customer GSOA*. This GSOA implements the R_{CS} reference point.

Adhering to the GSOA, distribution planes are used in the SDP architecture to provide services and applications with distribution independence. Also, distribution planes provide implementation, transport and technology independence. In the SDP architecture, the management GSOA is supported by the *Management Plane*. The network resource GSOA is supported by the *Network Plane*. The telco GSOA is supported by the *Telco Plane*. The customer GSOA is supported by the *Proxy Plane*. The open GSOA is supported by the *Open Plane*. The open plane provides external enterprises, such as application developers, content providers or customers, access to open GSOA applications and services. The GSOAs' distribution planes are interconnected via their services. This interconnection also contributes to the implementation of the SDP reference model's reference points.

7.5 Evaluation of SDP Business Model, Reference Model and Architecture

By extracting, extending and applying the generic SOA concepts to the SDP we have defined an enterprise service-oriented viewpoint. This viewpoint defines various abstractions that are structured in the SDP business model, reference model and architecture. The abstractions contribute to the definition of the SDP framework. We evaluate the SDP business model, reference model and architecture against the generic concepts extracted from the SOA. In addition, the evaluation provides answers to the questions posed in *Chapter 1 Section 1.3*. We also present the comparison between web service SOA, enterprise SOA, GSOA and the SDP architecture in *Table 7.1*.

The SDP business model enables the identification of business entities that interact with the SDP. These business entities are diverse and include various IT-using enterprises, customers and other telcos. For example, the business model may cater for fixed and mobile customers and telcos. In addition, IT-using enterprises may include service brokers, content brokers, connectivity providers and device manufacturers. Hence, the SDP business model is generic and may be easily *extended* to satisfy a telco's SDP requirements.

The SDP reference model is used to specify interactions between the various business model entities. These interactions are defined by decomposing business entities and business relationship points. As a result, the reference model defines business objects and reference points that *formalise* interactions between business entities. The reference points are important since they promote the *standardisation* of interactions between the business objects. For the SDP we motivate the use of services and their interfaces to implement reference

	Web Services SOA	Enterprise SOA	GSOA	SDP
Business Model	Web Services based		Inherits generic concepts from the web services business model.	Based on SDP requirements.
Reference Model	Derived from web services business model.		Inherits generic concepts from the web services reference model.	Based on decomposition of SDP business model.
Services	Web services based		All components of the architecture are technology and implementation neutral.	
Interfaces	WSDL based			
Applications	BPEL and others			
Architecture	Based on web services		Technology Neutral	Integration of multiple GSOAs.
Layers	Application	Abstracted by middleware	Used as a building block to structure layers	open SOA
	Service			
	Generic			telco SOA
	Function			network SOA
	Resource			
Domains	Derived from business entities.	Abstracted by middleware	Used as a building block to structure domains	Customer and Management SOAs.
Middleware	SOAP and HTTP	Uses ESB-based technologies	Distribution planes	Uses GSOAs with integrated distribution planes.

Table 7.1: Comparison of SOA the SDP Architectures

points.

The SDP architecture provides a *service-oriented perspective* on the SDP by defining services with interfaces to implement reference points. The architecture defines numerous services that abstract the abundant telco infrastructure. For instance, management, customer, service platform and network-oriented services are defined. Each service provides an interface to its functionality. These interfaces hide the service implementation and distribution details. As a result, diverse application implementations may orchestrate distributed service interfaces to provide specific functionality. The SDP architecture groups applications and each type of service into their own GSOA. Each of the GSOAs used are technology neutral. The SDP architecture also inherits this technology neutral property.

The SDP architecture does not use layers and domains as a design pattern to structure its services; rather it uses multiple GSOAs. However, the SDP architecture uses its GSOAs to abstract layers and their distribution across domains. For instance, the open, telco and network GSOAs show the layered hierarchy of service abstractions. Also, the customer and management GSOAs abstract the distributed customer and OSS/BSS domains. Communication across layers and domains is abstracted by the GSOAs service interfaces and their interconnected distribution planes. Thus, the GSOA provides an abstraction that is used to structure layers and domains for the SDP architecture.

Another benefit of using the GSOA to structure the SDP architecture is its distribution plane. The distribution plane contributes middleware functionality to the SDP architecture. Each of the GSOAs that constitute the SDP architecture, use distribution planes to abstract the distribution of applications, services and telco infrastructure functions. In addition, the distribution plane aids in hiding the implementation of applications and services from each other. The distribution plane also hides the underlying technologies that implement the communication mechanisms required to link applications, services and telco infrastructure.

The SDP architecture presented here drastically extends the proposed SDP architecture shown in *Figure 1.3*. For instance, the SDP architecture is technology neutral and uses service interfaces to promote SDP standardisation. These service interfaces also promote standard-based interworking between telco and external IT-based infrastructure, such as telco OSS/BSS and application developers SCE/SME. Also, the use of GSOAs provides a highly abstract structure for the SDP that is decomposable into layers, domains and even technology specific platforms.

7.5.1 SDP offering a Web Services SOA

Providing IT-using enterprises with applications and services is an untapped revenue stream for the telco [12]. Consequently, a requirement for the SDP is to promote the integration between enterprise and telco infrastructures, that is, support telecom-IT convergence. Hence, the SDP plays a role in generating revenue from enterprises for the telco. To enable this integration, the SDP offers one of its GSOA's to enterprises. This offered GSOA is the open GSOA. As a result, enterprises may outsource application or service development tasks to the telco. These applications and services are hosted by the SDP. Thus, the telco benefits from promoting itself as a SOA compliant outsourcer to enterprises [119]. The open GSOA must be implemented using web services, such that IT-using enterprises easily integrate with the telco infrastructure. However, the lower GSOAs of the SDP may be implemented using various other technologies. Thus, the open GSOA hides the underlying implementation of the SDP and telco infrastructure from these IT-using enterprises.

Figure 7.7 summarises the above scenario of SDP and SOA integration. The figure illustrates the SDP integrating the legacy IN *Service Control Point (SCP)* and new solutions operating over packet based telco networks. Also, the SDP provides an “enterprise SOA-like”

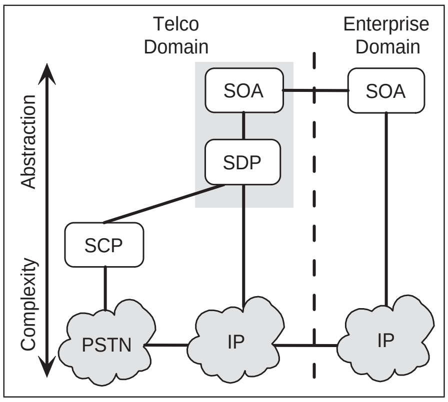


Figure 7.7: Telco and Enterprise Convergence

environment that enables telco and enterprise integration. The figure illustrates integration of networks being most complex, since they require integration via protocols. However, the figure illustrates integration of SOAs being least complex, since they use simple software-based web services, web service interfaces and ESBs. The figure also illustrates the increase of abstractions from networks to SOA. These abstractions represent services that simplify access and usage of telco infrastructure functions.

7.6 Summary

In this chapter we presented contributions of current a enterprise platform standard to the SDP and its framework. We reused generic concepts from the SOA standard. These concepts where extracted from the web services and enterprise SOAs. By reusing these concepts we defined a SDP business model, reference model and architecture. In both SDP business model and reference model we separated telco infrastructure according to its functions. We also defined reference points to formalise integration of these functions with external IT-using enterprises. We defined a SDP architecture from both SDP business model and reference model. The SDP architecture represents the collective of diverse services. Each of the services expose interfaces that implement reference points. To structure the architecture's services and interfaces across layers and domains, we use the GSOA. We defined the GSOA by extracting technology independent concepts from the various SOAs, such that implementation technologies and distribution mechanisms may be chosen and not imposed. The GSOA represents containers for services and applications. Applications orchestrate service interfaces. The GSOA abstracts service and application distributions by using its distribution plane as middleware. Therefore, by reusing generic SOA concepts we presented technology neutral abstractions and an architecture that contributes to the SDP framework.

Chapter 8

Perspective on the SDP from a Converged Standard: IMS

The telco network is a complex and distributed mass of transport links, service platforms, management systems and business solutions. Many network parts are implemented using telecoms' standards and technologies. Also, network parts may be proprietary solutions obtained from vendors. With progressive changes in technologies, standards, vendor solutions, customer requirements and telco business requirements the telco network continuously evolves. This evolution occurs within stages that define specific requirements. The result of satisfying each stage and its requirements is the decomposition of the telco network into various functional entities [120]. The evolution of the telco network is illustrated in *Figure 8.1*.

The telco network started with processor controlled PSTN switches. The switching hardware was tightly couple with service logic. This coupling limited service creation and provision. As a result, the first evolution stage required quick service creation and provisioning. As a result, the IN and CAMEL standards are defined. These standards define a distributed service platform, containing generic and reusable service building blocks used in service creation. Also, the standards define network functions required to support these building blocks. The network functions abstract the underlying network resources and capabilities.

The second stage aims to overcome IN/CAMEL limitations, so as to further improve service creation and provisioning. As a result, standards-based service platforms such as TINA were defined. TINA defines generic reusable software-based components that are used for customer service development. Also, TINA offers managed access of their components to external 3rd parties. Thus, both telco and external partners may host customer services. TINA components operate across middleware that was implemented by computing platforms. These computing platforms abstract access to network resources and capabilities.

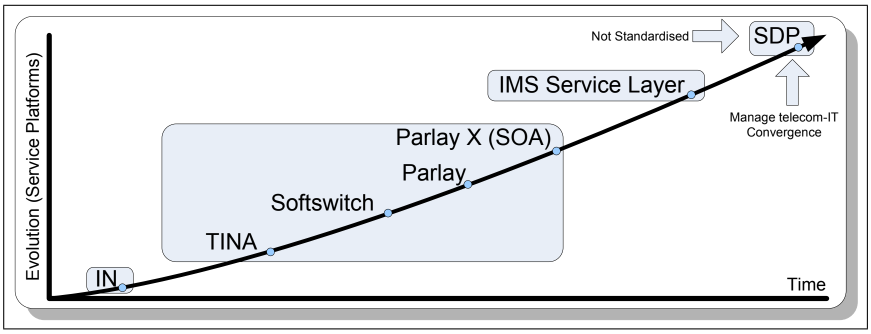


Figure 8.1: Evolution of Telco Network

The third stage aims to support interoperability between heterogenous transport networks, such as telco transport networks and the Internet. To fulfil these requirements, the softswitch standards are defined. The softswitch decomposes the traditional switch into functional entities that promote protocol and media conversion between telco networks and the Internet. For example, telephony services can originate and terminate on both telco network and Internet. The softswitch functional entities also promote the development of standards-based service platforms, such as Parlay and Parlay X. Services supported by service platforms may use softswitch capabilities to invoke common telco network facilities. Also, by using softswitch capabilities services may be delivered across telco networks and the Internet.

The fourth stage requires the convergence between telco, Internet and IT-based enterprise networks and services. As a result, the telco deploys packet-based networks that incorporate standard-based Internet protocols, such as IP. Also, [121] defines the *IP Multimedia Subsystem (IMS)* [24] standard. The IMS further decomposes the telco network and softswitch and introduces new functional entities that communicate using standard Internet protocols. The functional entities support new, old and current service platforms, such as SIP application servers, IN/CAMEL and Parlay gateways.

The current evolution stage aims to fulfil previous stages' requirements but centres on service platforms, like the SDP, that access and use the IMS network functions. Hence, the current evolution stage aims to structure and standardise the SDP while reusing existing network standards. In the following sections we discuss the IMS and its contribution to the definition of the SDP and its framework, with the objective of uncovering abstractions that constitute a technology neutral SDP architecture.

8.1 Requirements

The IMS is standardised by [121] and focuses on the evolution of the mobile telco and its network functions into a multimedia communication system. The IMS also evolves network functions to support mobile telco and Internet interoperability by using Internet protocols. The network functions support call/session signalling, transport network interworking, resource management and invoking service platforms. The network functions also support the delivery of IP multimedia services across the mobile telco's packet-switched network. However, by removing some of the network functions and their terminal mobility capabilities the IMS may be used in the fixed telco network. Thus, the IMS is applicable across both fixed and mobile telco networks.

The IMS is a packet-based network that is overlayed onto existing packet bearer networks. It integrates with existing telco networks, such as the General Packet Radio Service (GPRS) access network. It uses the GPRS network functions to enable customer access to the IMS. The IMS decomposes most of the remaining telco network functions, to offer more specific functionality. For example, the IMS extracts capabilities from the softswitch and defines additional signalling and media gateway functions. The IMS supports telecom-Internet interoperability by integrating Internet protocols into telco network functions. For instance, SIP, Diameter, HTTP and other Internet protocols are used by the telco network functions.

The IMS provides functions to support customer mobility between various circuit-mode networks and packet-based networks that are controlled by different network operators. The IMS functions also support service, customer and network signalling. Other limited IMS mechanisms contribute to end-to-end quality of service negotiations, charging, security and customer service subscription management. Thus, the IMS is considered a signalling overlay network that provides functions contributing to the overall operation of the telco network and delivery of services to customers.

The IMS aims to satisfy customer, service and network requirements [122]. For instance, the IMS supports customer access, registration and mobility. Also, service requirements include subscription management, access control, session control, service interworking and addressing. Network requirements include supporting service requirements, customer mobility and network interworking. Other requirements and IMS properties, as defined in [123], include:

- logical separation of signalling transport from bearer transport;
- providing multimedia services using Internet applications, services and protocols;

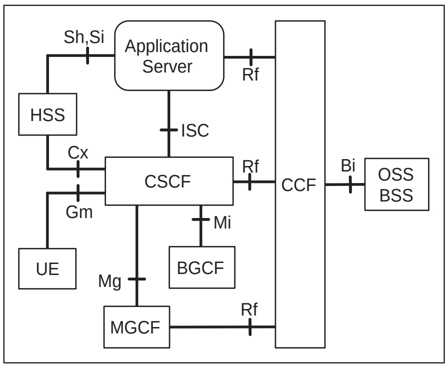


Figure 8.2: Simplified Portion of the IMS Reference Model

- non standardisation of customer services, rather customer services are developed by numerous external developers;
- multimedia services based on session control over IP;
- logical and physical separation of domains, such as home and visited network domains;
- physical mobility management provided by the access network, while the IMS components manages mobile users seamless access to their home network services;
- provide application servers that are internal and external to the telco; and
- online and off-line charging, where a customer may be billed based on various aspects of the service usage, such as service type, session period, media usage and terminating party.

As described in [24], the IMS defines various functional entities by decomposing telco network elements and integrating them with Internet technologies. Some of the functional entities are shown in *Figure 8.2*. We describe these and other functional entities in *Section 8.2.1*. The functional entities abstract complexities of using legacy and new transport networks, signalling networks, data stores, service platforms and OSS/BSS. In addition, these functional entities communicate amongst each other to fulfil IMS requirements. The collection of these functional entities and their interactions are structured within a reference model. The reference model formalises complex relationships between functional entities using *reference points*. The full IMS reference model with reference points is given in [123]. A simplified portion of this reference model with reference points is shown in *Figure 8.2*. With decomposition of the functional entities and reference points, IMS architectures are defined.

8.2 Architecture

We represent an IMS architecture using two separate models. The first architecture, shown in *Figure 8.3*, structures all the IMS functional entities defined in [123]. The functional entities include those shown in *Figure 8.2*. The second architecture, shown in *Figure 8.4*, structures a service platform architecture for the IMS.

8.2.1 Functional Architecture

The IMS functional architecture illustrates the various functional entities that are required to implement the IMS. Also, the architecture implements the reference model's reference points using standardised telecom and Internet protocols. Examples of protocols include SIP, Diameter, Megaco [41] and Real Time Protocol (RTP) [36]. These protocols are used for call/session signalling, authorisation control, bearer control and streaming media respectively.

In *Figure 8.3*, we use domains to illustrate the distributed nature of the IMS. These domains are the customer, partner and telco domains. Functional entities operate across these domains. For instance, SIP-based user equipment (UE) signals directly to a proxy call session control functional entity (P-CSCF), that manages customer-to-IMS signalling. Additional functional entities are defined to administer the interworking between converged transport networks, that is, the border gateway control function (BGCF). Also, entities control media, signalling, protocol and transport interworking between circuit and packet-based networks. These are the media gateway control function (MGCF), signalling gateway (SGW) and media gateway (MGW). The delivery of media across transport networks is also controlled by the media resource function controller (MRFC) and media resource function processor (MRFP). Functional entities, such as the online charging function (OCF) and charging collection function (CCF), are used to abstract the telco OSS/BSS charging capabilities. These functional entities are used by various other entities to report and obtain billing information. All functional entities communicate using the SIP or Diameter protocols. Diameter is used for communication with the telco OSS/BSS and home subscriber server (HSS) functional entities.

In the IMS, two functional entities are used to support various service platforms. These entities are the HSS and serving call session control function (S-CSCF). The HSS is a central data management function that administers service and network-related data. Examples of HSS data include legacy mobility management data and per user IN service type data. Hence, the HSS represents a converged service data source within the IMS. The S-CSCF

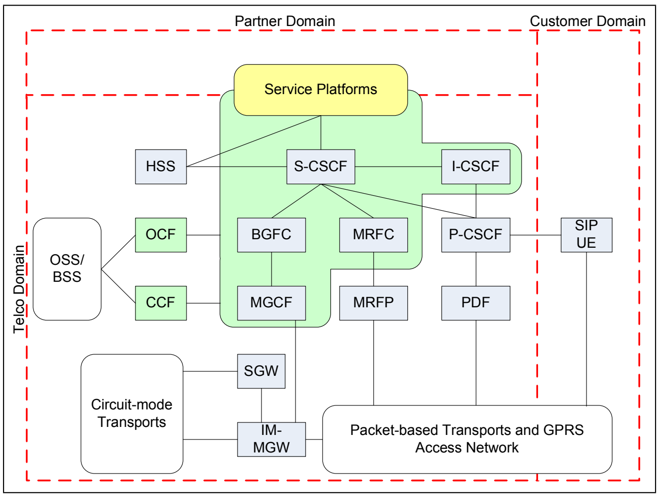


Figure 8.3: IMS Functional Architecture

provides limited intelligence for the IMS at the call-session signalling level. It satisfies SIP requests from customers by interacting with various SIP application servers.

IMS functional entities represent abstractions of telco network functions. Functional entities implement reference points, using protocols, so as to enable access to network functions. As a result, functional entities are used in developing and delivering customer services. These customer services are defined, developed, deployed and managed within an IMS service platform architecture.

8.2.2 Service Platform Architecture

Unlike the functional architecture, the IMS does not define a fully structured service platform architecture. Rather, it proposes a limited *service layer* [124]. The service layer contains a collection of *SIP Application Servers (SIP AS)* that host SIP-based applications. SIP applications process SIP messages obtained from functional entities. Also, SIP applications generate SIP messages intended for the functional entities. The service layer uses SIP to implement the ISC reference point shown in *Figure 8.2*. Some SIP applications also communicate with functional entities using Diameter and CAMEL related protocols. These protocols implement the Sh and Si reference points also shown in *Figure 8.2*.

The IMS service layer abstracts access to the HSS and S-CSCF functional entities by providing a foundation of functionality, that is used by service platforms. The IMS standards prescribe only three service platforms that can be used: a SIP-based service platform, a CAMEL service environment and a Parlay Gateway. SIP-based service platforms contribute additional SIP application servers to the IMS service layer. Applications contained in these servers provide a variety of services to customers. For the IMS the CAMEL service environment delivers traditional voices services to customers. The standards produced by [121] define intermediate layers that enable translation between CAMEL protocols and SIP. The Parlay Gateway supports internal and external application servers that host numerous customer services. Applications contained in these servers access Parlay Gateway SCFs via their APIs. The standards produced by [121] also define intermediate layers that enable SCFs to use the IMS service layer capabilities. Thus, the CAMEL service environment and Parlay Gateway use their intermediate layers to simplify the ISC, Sh and Si reference point implementations.

Based on the IMS standards for the three types of service platforms ([125, 126, 127, 128, 129]), we synthesise an IMS service platform architecture. This architecture is illustrated in *Figure 8.4*. The figure illustrates the IMS functional entities, such as those shown in *Figure 8.2*, towards the bottom layers. The IMS service layer is depicted as SIP application servers above the functional entities. Also, the figure shows the ISC, Sh and Si reference points being implemented as SIP, Diameter and CAMEL protocol communication between service layer and functional entities. Crosses in the figure illustrate non-standard reference points implemented with non-standard protocols. The remaining parts of the figure express the richness of the service platforms that constitute the IMS service platform architecture. For instance, grey boxes represent components that house applications, while yellow boxes represent service functions.

In *Figure 8.4*, the service layer interworks with CAMEL and Parlay service platforms. To support integration with the CAMEL, an *IP Multimedia Service Switching Function (IM-SSF)* is defined. The IM-SSF contains a SIP application server that manages conversions between SIP messages and basic call state model detection points. A modified CAMEL service switching function (imcnSSF) is also defined to perform detection point processing. The communication between SIP application service and imcnSSF is not standardised. The imcnSSF communicates with a modified CAMEL service control function (gsmSCF) that hosts application logic used to provide customer services. The gsmSCF resembles an application server that is accessed via the IMS functional entities.

The IMS service platform architecture also shows the service layer integrating with a Parlay service platform named the *Service Capability Server (SCS)*. The SCS contains *Service Capability Features (SCF)*, SCS-logic and a SIP application server. The SIP application server enables Parlay and IMS interworking by converting between SIP messages and SCS-logic requests. However, this communication is not standardised. The SCS-logic also integrates with the CAMEL gsmSCF, by converting between SCS-logic requests and CAMEL protocols. However, this interworking is also not standardised.

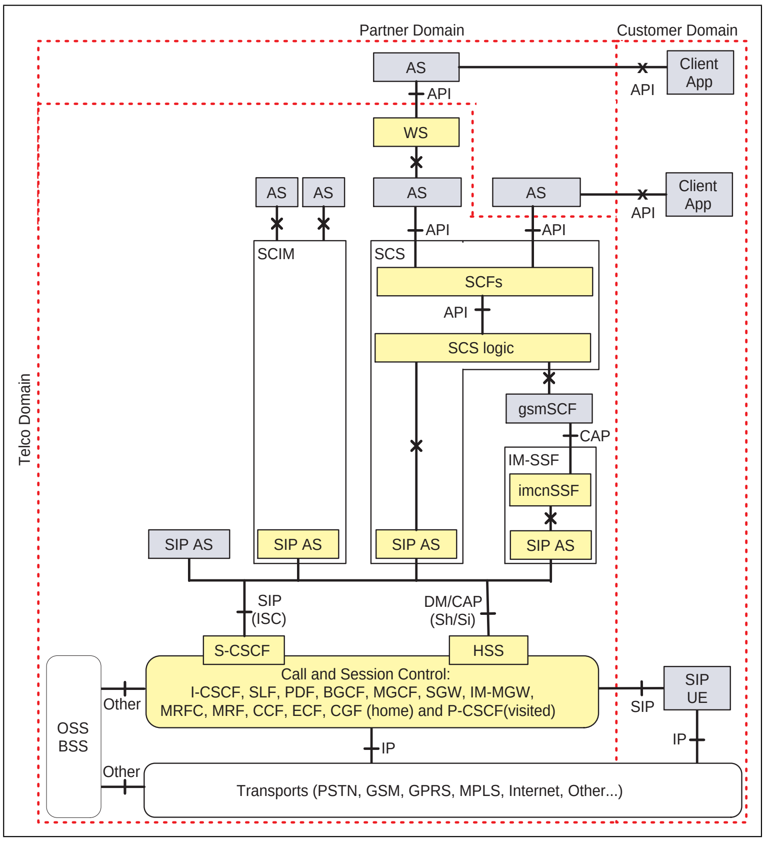


Figure 8.4: IMS Service Platform Architecture Synthesised from IMS Standards

In *Figure 8.4*, standard APIs are used between the SCS-logic, SCFs and Parlay-based application servers. These APIs are technology neutral and hide implementations of the SCS parts. The APIs offer telco and external partners standardised access to SCFs, that hide details of the IMS functional entities [123]. APIs are used by client applications residing in Parlay-based application servers. Thus, APIs enable easier customer service development by promoting reuse of SCFs. Client applications, in the customer domain, also use an undefined API to interact with Parlay-based application servers located in both telco and partner domains. To support the distributed use of the APIs a distribution mechanism is used, such as CORBA.

By using Parlay, a Parlay X service platform can be incorporated into the IMS service platform architecture. As illustrated in *Figure 8.4*, the Parlay X service platform contains a collection of web services (WS). Web services simplify access to SCFs using standardised implementation independent APIs. These APIs are exposed to external partner applications residing on Parlay X based application servers. Web services use the SCFs by invoking Parlay-based applications. The communication between web services and Parlay-based applications are not standardised. Also, client applications in the customer domain use an undefined API to interact with Parlay X based application servers. Both standard and undefined APIs are supported by web services-based protocols, such as SOAP and HTTP.

In addition, ESBs may be used to hide the distribution of the Parlay X web services and applications.

The IMS standards identify an additional mechanism used to access and deliver customer services. This is the *Service Capability Interaction Manager (SCIM)*. The SCIM is included in the IMS service architecture shown in *Figure 8.4*. Though not fully defined, the SCIM aims to manage interactions between diverse application servers [125]. Therefore, we assume the SCIM orchestrates multiple application invocations across multiple application servers, so as to provide a customer service. For example, [130] uses the SCIM to orchestrate multiple SIP application invocations. Also, [131] uses the SCIM to translate web service invocations into multiple SIP application invocations. This is not shown in *Figure 8.4*. We assume the application servers managed by the SCIM are of various types and within the telco domain. Similarly to other service platforms, the SCIM interworks with the functional entities using a SIP application server. However, due to the limited SCIM definition, communication between application servers is undefined. Also, customer services hosted across the application servers are accessed by customers via the functional entities.

The IMS also uses stand-alone SIP application servers to host SIP-based applications. These SIP application servers are contained within the telco domain and their applications use the functional entities directly. Customers access their services, provided by SIP applications, via the IMS functional entities. In the IMS service architecture SIP application servers providing customer services are included in the IMS service layer.

The IMS service platform architecture represents a complex mesh of SIP-based application servers and various service platforms. These service platforms are standards-based. However, the IMS service architecture identifies non-standardised interfaces between the IMS service layer and the various service platforms. In addition, non-standardised interfaces are identified within some of the service platform architectures. By abstracting the complex structure of the IMS service platform architecture and its non-standardised interfaces we can extract generic concepts that are applicable to the SDP and its framework. In a similar approach concepts can be extracted from the IMS reference model and functional architecture.

8.3 Reusable Concepts

The reference model in [123] and summarised in *Figure 8.2* provides a *functional perspective* on the IMS. This functional perspective focuses on adapting Internet-based call and session control capabilities for use within the telco network. As a result, functional entities are defined to support these new capabilities within the telco network. These functional entities provide a range of functions such as customer, network and service oriented functions. Some functional entities are borrowed from existing Internet-based architectures and deployed in the telco network. However, new functional entities are also required. These functional entities *abstract* new or existing telco network functions.

The reference model in [123] shows the complex interactions between functional entities required to satisfy IMS requirements. In the reference model functional entity interactions are formalised using *reference points*. The reference points formalise the relationships between functional entities and promote their standardisation. Reference points are abstract and implementation, technology and distribution independent. For example, reference points may define system viewpoints that produce architectures. Alternatively, reference points may define implementations using specific protocols or APIs. Thus, reference points are used to promote the *standardisation* of any relationship.

Our functional architecture shown in *Figure 8.3* also provides a *functional perspective* on the IMS. In this perspective we organise the structure of the IMS reference model to illustrate the interactions of functional entities across *domains*. These domains indicate the distribution of functional entities across the telco network and customer terminals. Also, a partner domain is used to indicated external access to service platforms that use IMS functional entities. The functional architecture also implements reference model reference points using protocols such as SIP and Diameter. These protocols enforce standardised communication between functional entities. In addition, these protocols support the new call and session control capabilities gained by the telco network.

The IMS service layer provides a limited *service-oriented perspective* that aims to link functional entities with various service platforms. Our service platform architecture provides a more complete service-oriented perspective on the IMS by showing what is possible between the IMS service layer and various service platforms. We decompose the IMS service layer and the various service platform architectures to contain generic reusable *services*, that abstract access to the IMS functional entities. These services represent reusable building blocks used in application development. The execution of these applications provide customer services. The services are diverse and provide a range of functionality. Like the functional architecture the service platform architecture formalises interactions between the various services and applications. These interactions are implemented using standard-based *interfaces*. The interfaces are less complex than network protocols and defined as implementation and distribution independent APIs.

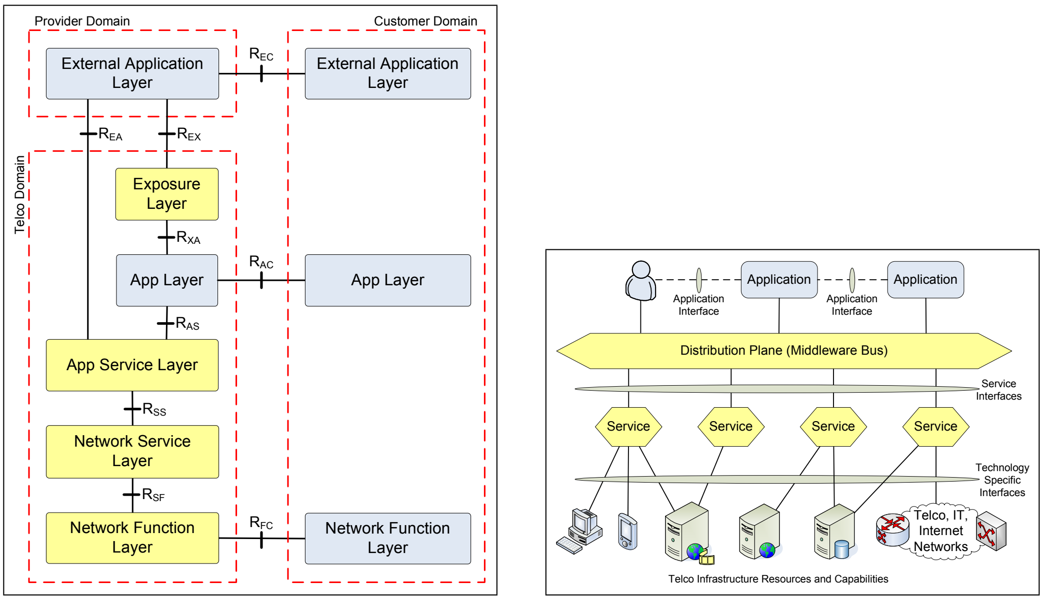
We extend the service-oriented perspective on the IMS by structuring our IMS service platform architecture using *layers* and domains. The various service platforms that constitute the IMS service platform architecture are decomposed into numerous layers. Layers contain either services or applications. The lower layers closest to the functional entities contain services that abstract access to functional entities via their interfaces. Higher layers contain services with interfaces that abstract access to lower layer services. These higher layer services offer their interfaces to internal and external applications contained in an application layer. Both application and service layers are distributed across domains. These domains are equivalent to those defined in the IMS functional architecture.

We identify another reusable concept from our IMS service platform architecture. This concept being *middleware*. Service interfaces are implemented as APIs that operate across distributed domains. For example, Parlay SCFs are invoked by applications contained in the partner domain. The use of middleware ensures that distribution is hidden from external applications invoking service via their interfaces and services invoking each other via their interfaces. Also, Parlay SCFs provide a form of middleware since their APIs hide details of the ISC, Sh and Si reference point implementations. Therefore, the SCF APIs represent service-oriented capabilities that make use of the network-oriented IMS functional entities.

8.4 Contribution to the SDP from the IMS

By reusing and applying the generic IMS concepts we present a SDP reference model and architecture. The SDP reference model is shown in *Figure 8.5(a)*. The SDP reference model is derived by generalising the IMS service platform architecture. We use layers, domains and reference points to structure the reference model. As a result, the reference model also illustrates the SDP within its environment.

In the SDP reference model, layers are used as a primary abstraction tool to group and *structure* applications, services and network functions contained across the SDP and telco. We generalise the Parlay and Parlay X application server used in the IMS service platform architecture as an external application layer. This layer contains 3$^{\text{rd}}$ party applications created by application providers. These applications use services offered by the exposure layer or application service layer to fulfil their requirements. The exposure layer contains generic services that enable the basic telco network capabilities to be invoked. Thus, the exposure layer abstracts the Parlay X Gateway. The exposure layer services use services contained within the application layer to invoke telco network capabilities. The application layer also contains applications. This layer's services and applications use the complex services offered by the application service layer to fulfil their functions. Thus, the application layer abstracts the capabilities provided by a Parlay application server. The application service layer contains complex services that enable the full capabilities of the telco network to be



(a) SDP and its Environment derived from IMS Service Platform Architecture

(b) GSOA Detailing Generic Concepts

Figure 8.5: SDP Models based on IMS Concepts

invoked. Thus, the application service layer generalises the Parlay Gateway.

The network service layer represents a *point of integration* between the various higher application and service layers and the underlying network capabilities and resources. As a result, the network service layer provides services that invoke the appropriate network functions. Thus, the network service layer abstracts the IMS service layer. The network function layer contains the various telco network functions. These functions may also be located on the customer device. Therefore, the network function layer may contain call and session control functions similar to those offered by the IMS functional entities.

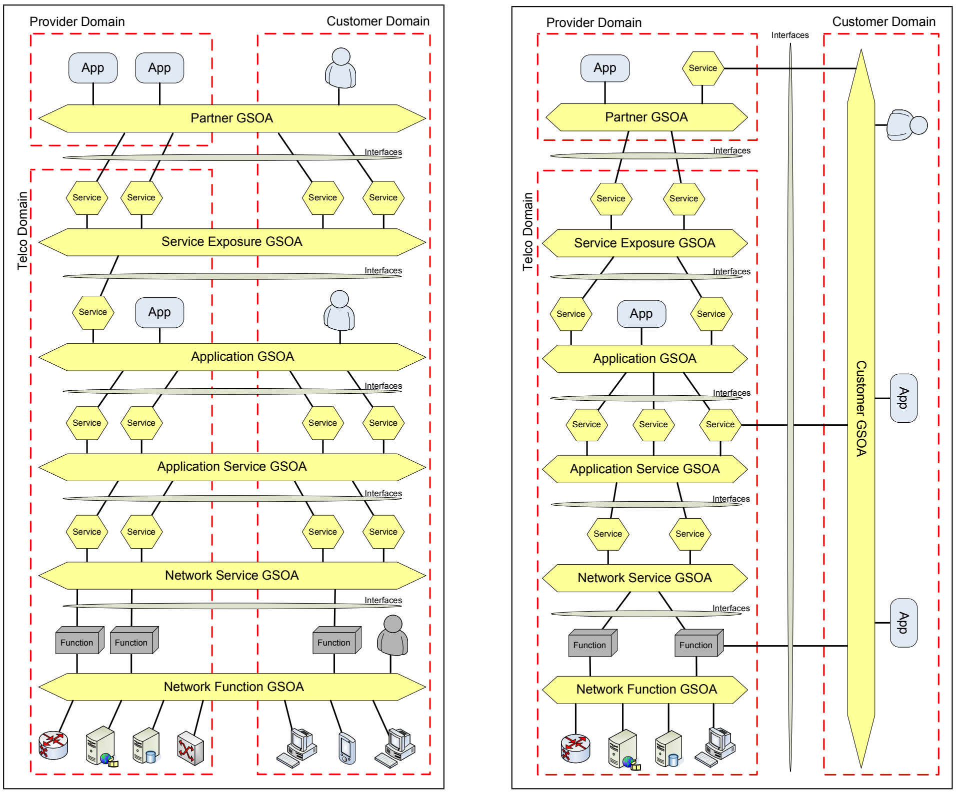
The reference model *distributes* layers across various functional areas by using domains. These domains specify areas of interest that services must operate within. Domains prescribe additional decomposition of layers to include distributed communication of services. The domains shown in *Figure 8.5(a)* are equivalent to those used throughout the IMS reference model, functional architecture and service platform architecture. The customer domain represents the customer device or terminal, that enables the customer to access and use the various applications and services offered by the SDP. The partner domain represents an external 3rd party. For example, partners include application provider enterprises that require access to telco network capabilities and resources to deliver content or create customer services. The telco domain represents the area in which the SDP operates. In this domain, the SDP has access to the various telco network functions.

Within the SDP reference model, reference points promote *standardised* communication between service and functional abstractions, that are contained within layers and distributed across domains. For instance, the R_{EA} and R_{EX} reference points formalise communication between the telco and external IT-based infrastructures. The R_{EC}, R_{AC} and R_{FC} reference points standardise communication between the customer, applications and the network. The R_{XA}, R_{AS} and R_{SS} reference points promote standardised communication within the SDP and between its various services. Also, the R_{SF} reference point defines communication between the SDP services and converged network functions. This reference point demands greater standardisation than the current ISC reference point. By decomposing or implementing the various reference points, additional service-oriented details on the SDP are revealed. Hence, a SDP architecture can be defined.

To structure the SDP architecture we reuse the GSOA concept defined in *Chapter 7*. As determined, the GSOA is used to define service platform architectures independent of technology, distribution and implementation details. This is achieved by using the GSOA as a reusable building block that manages various services, applications, interfaces, infrastructure and their distribution. Hence, multiple GSOAs are used to manage the various customer applications, partner applications, SDP applications, SDP services, network functions and all corresponding interfaces. In addition, the GSOAs are used to structure the SDP reference model's layers and abstract the distributed domains using their distribution planes. Hence, multiple GSOAs decompose the SDP reference model's reference points and support the communication of application, services and functions between layers and across domains, using their interfaces.

We illustrate the GSOA, used for decomposing the SDP reference model, in *Figure 8.5(b)*. This representation is similar to the GSOA shown in *Figure 7.4*. In the figure, applications provide customer services, via their interfaces. Also, numerous service interfaces are defined and easily accessible by applications. Services access underlying infrastructure resources and capabilities via infrastructure interfaces. To enable technology, implementation and distribution independence of application, service, and infrastructure interfaces, the distribution plane is used. However, in this GSOA representation the distribution plane provides a technology neutral middleware bus. The middleware bus contains its own middleware services that are accessible to applications, services and infrastructure via interfaces. Some of these middleware services may be used to simplify access to infrastructure functions. As a result, the middleware's services and their interfaces represent the abstraction of infrastructure resources and capabilities.

A SDP architecture structured using GSOAs is shown in *Figure 8.6(a)*. In the architecture, layers are represented using multiple GSOAs. Since layers are hierarchically structured, their associated GSOAs are also layered and simplify access to each other. This illustrates



(a) SDP Architecture (b) Alternative SDP Architecture

Figure 8.6: SDP Architectures based on Generic IMS and SOA Concepts

GOSA applications, services and middleware abstracting other GSOAs. The GSOAs with their service interfaces implement the various reference points. The service exposure GSOA implements the R_{EA} and R_{EX} reference points. By implementing the R_{EA} reference point, the service exposure GSOA uses its service interfaces to simplify access to complex services hosted in the application service GSOA. The application GSOA implements the R_{XA} reference point, while the application service GSOA implements the R_{AS} reference point. The network service GSOA and network function GSOA implement the R_{SS} and R_{SF} reference points respectively. The layered GSOAs with the aid of their service interfaces also implement the R_{EC}, R_{AC} and R_{FC} reference points into the customer domain.

Since the GSOAs are generic, they may be used to restructure the SDP architecture. An alternative SDP architecture, shown in *Figure 8.6(b)*, illustrates the use of GSOAs within domains. The figure shows a unified customer GSOA with services that abstract resources and capabilities found within the customer domain. Similar to the properties of the GSOAs in *Figure 8.6(a)*, the customer GSOA services abstract other GSOAs, their services, applications and functions found within the provider and telco domains. Hence, the customer GSOA and its service interfaces implement the R_{EC}, R_{AC} and R_{FC} inter-domain reference points.

8.5 Evaluation of SDP Reference model and Architecture

By applying the generic IMS concepts, in particular the service platform options, to the SDP we have defined functional and service-oriented viewpoints. Both viewpoints contribute abstractions that are structured in the SDP reference model and architecture. The abstractions contribute to the definition of the SDP framework. We evaluate the SDP reference model and architecture against the generic concepts extracted from the IMS. In addition, the evaluation provides answers to the questions posed in *Chapter 1 Section 1.3*. We also present the comparison between the IMS architectures and the SDP architecture in *Table 8.1*.

We do not define a SDP business model using the generic IMS concepts. However, the IMS derived SDP reference model incorporates some business model concepts and properties. The SDP reference model encapsulates the requirements of the SDP. Examples of these requirements include, supporting customer service development and management by providing generic services to various external IT-using enterprises. These SDP services must support customers in locating, registering, consuming and paying for customer services. Like a business model, the reference model defines relationships between external partners, customers and the SDP. However, the reference model promotes standardised communication between SDP, customers and external partners using *reference points*. In addition, the reference model does not enforce specific technologies to implement reference points.

The SDP reference model provides a *functional perspective*, since it aids the definition of SDP functions. These functions are represented as a collection of applications, SDP services and network functions that use their interfaces to implement reference points. In this perspective, the reference model uses *layers* to structure the various SDP services, applications and network functions. The layers illustrate the relationship between the SDP services, applications and network functions. In addition, layers model the hierarchy of *abstraction* from complex network functions to simple services contained in the reference model's service exposure layer. Besides layers, the reference model uses *domains* to distribution applications, SDP services and network functions. Domains show applications, services and functions being invoked across various locations that are internal and external to the telco.

By decomposing the reference model into a SDP architecture we provide a *service-oriented perspective*. The SDP architecture manages the various applications, services and network functions within GSOAs. Thus, GSOAs represent the design pattern used to structure the SDP architecture. The GSOAs ensure technology, implementation and distribution independence. As a result, the SDP architecture remains technology neutral. The GSOAs fully implement the reference model's reference points without enforcing specific technologies.

	IMS	SDP
Reference Model	Defines functional entities and their interactions using reference points.	Defines collections of applications, services and functions contained in layers, distributed across domains and interacting via reference points.
Architecture Layers	Web service based application server	Partner GSOA
	Parlay X Gateway	Service Exposure GSOA
	SIP AS and Parlay Application Server	Application GSOA
	SCIM and Parlay SCS	Application Service GSOA
	IM-SSF and SIP AS	Network Service GSOA
	Network Functions (CSCFs, PDF, ...)	Network Function GSOA
	Network Elements	
Domains	Distributes layers across customer, external 3^{rd} party and telco functional areas.	
Middleware	ESBs	Provided by various GSOA middleware buses. These buses are technology and implementation neutral.
	CORBA	
	Protocols	

Table 8.1: Comparison of IMS and SDP Architectures

In addition, the GSOAs structure the layers and abstract the distributed domains using their distribution planes. Hence, application and service communication is fully managed and abstracted by the SDP using the GSOAs. The SDP architecture also shows higher layer GSOAs abstracting lower layer GSOAs. For example, lower GSOA service interfaces are abstracted by higher layer GSOA services. Also, the GSOAs supports service communication within a layer, via service interfaces.

Each GSOA provides a technology neutral distribution plane that is represented as a *middleware* bus. This plane abstracts technologies and hides distribution complexities from applications, services and functions. Also, the distribution plane also provides middleware services that are accessible via their interfaces. These middleware services may be used by SDP services to access network resources and capabilities. The GSOAs' distribution planes may be implemented using various technologies, such as web service based ESBs or CORBA. Hence, the GSOAs provide generic middleware functionality that ensures the SDP architecture remains void of any technologies.

The SDP architecture presented here overcomes many limitations of the proposed SDP architecture shown in *Figure 1.3*. For instance, we use functional layers that expose technology neutral interfaces to services or network functions. These interfaces consistently implement reference points and promote standardisation of the SDP. For example, some service interfaces provide access to network functions that abstract telco OSS/BSS capabilities. These service interfaces promote standardised access between telco OSS/BSS and external SME/SCE. We also use domains to separate the SDP across multiple functional areas. We include a customer domain such that customer access to the SDP is specified.

8.6 Summary

In this chapter we presented contributions of a current telco network standard to the SDP and its framework. We reused generic concepts from the IMS standard. Concepts where extracted from the IMS reference model and a derived functional architecture. We also synthesised an IMS service platform architecture that contributed all the generic service-oriented concepts to the SDP. By reusing these concepts we defined a SDP reference model and architecture. By removing technologies from the synthesised IMS service platform architecture, we defined a SDP reference model. This model contains applications, services and network functions that are structured within layers. The layers are also distributed across functional domains. Interactions between layers and across domains are formalised using reference points, to promote standardised communication. We defined the SDP architecture by elaborating the reference model, using the GSOA concept. We used multiple layered GSOAs to define the SDP architecture. These layered GSOAs structured SDP applications, services, functions and their interfaces. Lower layer GSOAs manage service (or network function) interfaces and provide higher layer GSOAs access to these interfaces. Together, GSOAs and their service interfaces implemented various reference model reference points. GSOAs were also used to abstract domains by using their distribution planes as middleware buses. GSOAs are generic, technology, implementation and distribution neutral. This ensured technology neutrality of the SDP from reference model to architecture. Therefore, by reusing generic IMS concepts with the GSOA we presented technology neutral abstractions and an architecture that contributes to the SDP framework.

Chapter 9

Defining the SDP Framework

In the previous chapters we have discussed telecom and enterprise standards. Each of these standards share similar service requirements with the SDP. The most prevalent requirement being the creation, delivery and management of customer services using telco and IT-based infrastructure. The standards that we have reviewed satisfy this requirement by defining concepts and abstractions to simplify the complexity of modifying or extending the telco and IT-based infrastructures. These concepts and abstractions include a range of business models, reference models and architectures.

The extraction of concepts from the TINA, Parlay, management and SOA standards have produced generic SDP business models. The business models demarcate the various telco and IT-using enterprise roleplayers that benefit from interacting with the SDP. Also, numerous business relationships are specified between the SDP and roleplayers. The SDP business models derived from generic TINA, TMN, TOM and eTOM concepts formalise business relationships and promote standardised communication between the roleplayers. SDP service interfaces are identified to provide standardised communication between roleplayers.

The SDP business models derived from Parlay and SOA concepts are further decomposed into SDP reference models. A SDP reference model is also derived from generic IMS concepts. The reference models use reference points to promote standardised communication between the SDP, telco-based infrastructure and external IT-based infrastructure. Reference points are elaborated into interfaces that are exposed by a variety of SDP services. Therefore, the reference model formalises the integration between telco and IT-based infrastructures by using service-oriented reference points.

The generic concepts obtained from the reviewed standards have contributed to the development of various SDP architectures. The SDP architectures are also based on corresponding business models and/or reference models. The architectures show the SDP contained within an environment that operates across various telco and IT-based infrastructures. The architectures illustrates layering of SDP services according to the level of abstraction their interfaces provide. Services are also distributed across domains, but distribution is hidden by middleware planes. All SDP architectures are technology neutral. However, the architectures promote the use of standards-based technologies to implement their service interfaces.

The management, SOA and IMS derived SDP architectures provide different approaches to structuring the SDP within its environment. The SDP architecture derived from TMN, TOM and eTOM concepts models abstractions required to manage the operation of the SDP and its services. The SDP architecture derived from SOA concepts shows the versatility of the GSOA to structure SDPs. This is also illustrated in the SDP architecture derived from generic IMS concepts. The IMS derived SDP architecture uses GSOAs to layer applications, services and middleware. Also, the GSOAs represent the elaboration of reference points found in the IMS derived SDP reference model.

By extending the previous chapters concepts and abstractions we define the SDP framework. We do this by revisiting the SDP definition and elaborating on key SDP requirements. The concepts and abstractions are used to satisfy the SDP requirements. As a result, we define a complete SDP business model and reference model. Also, we structure the SDP framework as an architecture illustrating the SDP within its environment.

9.1 Definition and Requirements

We define the SDP as a *"a distributed IT-based system that abstracts telco network capabilities into generic services that are accessible across telco, enterprise and Internet networks and promotes the development, delivery and management of various customer services"*. Using our definition we define the SDP requirements in the following sections. These requirements provide answers to the questions posed in *Chapter 1 Section 1.3*.

9.1.1 Infrastructure Integration

We need the SDP to simplify interworking between traditionally separate telco infrastructure systems. Also, we need the SDP to simplify telco and IT-based infrastructure interworking. Many vendor-based SDP products, such as [56], promote improved telco infrastructure interworking to streamline various telco activities. These products also provide proprietary solutions that interwork specific telco systems and external IT-based systems. However, in the previous chapters we have derived technology neutral SDP architectures that promote standardised infrastructure interworking. For instance, all derived SDP architectures provide layers of abstractions that simplify access to telco network resources and capabilities. These abstractions are exposed to IT-based systems.

All derived SDP architectures define and layer abstractions by first *separating* telco network functions from their complex physical representations. These network functions simplifies access to telco infrastructure parts. Some functions include setting-up connections, negotiating transport QoS, configuring device capabilities and updating billing databases. Separated functions are complex to use since they remain technology and distribution specific. To solve this problem, the SDP architectures abstract network functions into reusable, technology-neutral and distribution-neutral software-based *services*. These services provide functionality, such as make multiparty calls, obtain customer location, manage service profiles, send and receive messages and query customer accounts.

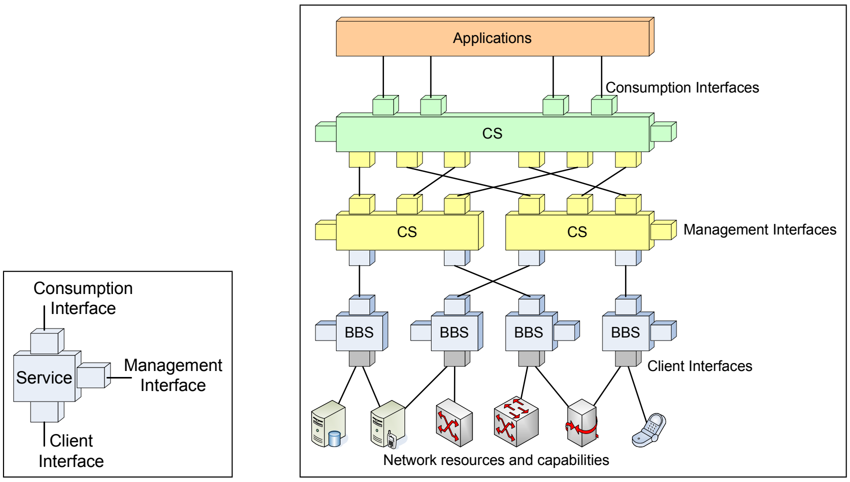
SDP services enable standardised access to all telco infrastructure parts for various activities, such as decreasing service development effort, improving network management and streamlining business and operational processes. In addition, SDP services are offered to external IT-based infrastructures. Enterprises may use SDP services to enhance their existing applications or create new applications. Thus, SDP services support integration between telco and IT-based infrastructures.

9.1.2 Service-oriented System

We need SDP services to expose their functionality to external IT-based infrastructure. Also, we need to classify the different types of SDP services that have been identified in the previous chapters' SDP architectures.

SDP services use generic mechanisms to offer access to their functionality. These mechanisms are called *interfaces*. Interfaces prescribe, in an implementation and distribution independent manner, what functionality a service offers. Therefore, applications using service interfaces are not constrained to specific implementation and distribution technologies. [8] and [55] recommend a service offer the following interfaces:

- consumption interface - exposes a service's available functionality.
- management interface - used for service administration.
- client interface - enables a service to use other services consumption and management interfaces.



(a) SDP Service Interfaces

(b) Types of SDP Services

Figure 9.1: SDP Services and Interfaces

We illustrate these interfaces in *Figure 9.1(a)*.

These interfaces have been used to access layers of services that constitute our derived SDP architectures. For example, in our SOA derived SDP architecture lower service layers offer consumption interfaces to higher layer applications and services. Though not specified in any of our derived SDP architectures, higher layer services may use client interfaces to access lower layer services on their consumption interfaces. In the TMN, TOM and eTOM derived SDP architecture we have illustrated layers of services that expose management interfaces. These services are invoked on their management interfaces by management services or management applications.

SDP services contain functionality that abstract complex infrastructure functions. Since numerous functions exist within the telco network, a number of services are created. Within the SDP, we define two categories of services: building block services and composite services. We illustrate these different types of SDP services *Figure 9.1(b)*. In the figure we show building block services as simplifying access to network functions. Building block services also perform distinctive tasks and do not interwork with other building block services. These building block services are used in our derived SDP architectures. For example, the Parlay derived SDP architecture shown in *Figure 5.6* uses generic services as building block services to simplify access to network functions.

Figure 9.1(b) shows composite services interworking with one or more building block services to further simplify access to infrastructure functions. In addition, composite services may provide new functionality by integrating multiple composite services and simplifying their use. The nested nature of composite services is finite. Thus, composite services are defined until no further simplification is gained or no unique functionality is defined. Like building block services, composite services are used in our derived SDP architectures. For example, the Parlay derived SDP architecture shown in *Figure 5.6* uses application services as composite services to simplify access to generic services.

9.1.3 Business Model

We need a business model for the SDP to accommodate diverse customers, external IT-using enterprises and other telcos.

The SDP sustains a business environment [132] that supports the convergence of various business entities. Within the business environment the SDP defines business relationships between diverse business entities. Business entities and their relationships are structured into a business model. In previous chapters we have derived numerous SDP business models. All the derived business models identify business entities that use the SDP for its services. These business entities are IT-using enterprises such as application developers and content providers. Also, some business entities include individuals or enterprise that access applications and content. As a result, we generalise these business entities as *customers* of the SDP.

In the SDP business environment customers play various business roles. These roles are extracted from the derived SDP business models in the previous chapters. Three common types of customer roles are provider, consumer and broker. The SDP itself plays the provider role since it provides services to consumers and brokers. In some cases the SDP may take on the consumer role since it consumes services offered by other providers. For example, the Parlay derived business model shows the SDP consuming services that are offered by external service providers. These services contribute to the SDP's own repository of services.

The provider role is also fulfilled by business entities such as application developers, content providers and the telco. These providers are prevalent in all the derived SDP business models. Application providers provision diverse applications to consumers. Content providers provide content to be used by applications. The telco provides SDP services with access to diverse network functions. The network functions expose telco transport capabilities and OSS/BSS capabilities. The application and content provider business entities also fulfil the consumer role. For example, application providers orchestrate SDP services to create applications, while content provides use SDP services to deliver content across the telco network. The telco may also fulfil the consumer role. For example, some network functions may invoke SDP services so as to notify the SDP of changes in the network.

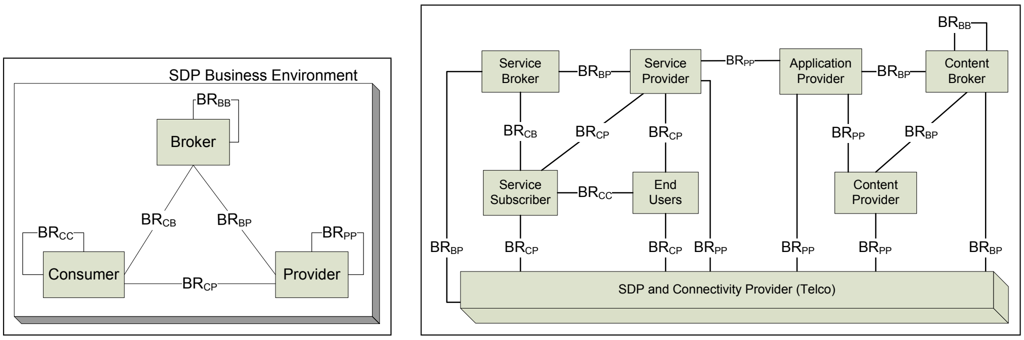
Other business entities that fulfil the consumer role include individuals or enterprises that act on behalf of individuals. These business entities subscribe, use and pay for applications that are managed or accessed via the SDP. These applications provide business entities with telephony, multimedia or data services. Business entities that take on the consumer role may also consume SDP services to subscribe, configure, access and use applications. In addition, these business entities may use SDP services to manage their personal profiles, service subscriptions and billable accounts.

Business entities that fulfil the broker role aim to support interactions between consumers and providers within the SDP business environment. Brokers are incorporated in various business models. For example, [53, 55, 54] and [130] promote the use of brokers with their SDP solutions. We have also used brokers in our TINA, Parlay, SOA and eTOM derived SDP business models. In these derived SDP business models we have used the SDP business entity as an all purpose broker. The objective of the broker is to use SDP services to create applications. These applications differ from other applications, since they offer functionality to both consumers and providers. For example, consumers use broker applications to find applications offered by application providers. This example implies application providers have already used broker applications to register their applications and intend to provide these applications to consumers.

A generic SDP business model is shown in *Figure 9.2(a)*. The generic SDP business model condenses the previous chapters' SDP business models by showing relationships between customers that fulfil the consumer, provider and broker roles. The relationships between customers are structured using *business relationship points*. The concept of business relationship points has been reused from TINA, Parlay, SOA and eTOM derived SDP business models. The business relationship points in the generic SDP business model are:

- BR_{CB} - consumer to broker relationship.
- BR_{BP} - broker to provider relationship.
- BR_{CP} - consumer to provider relationship
- BR_{CC} - consumer to consumer relationship.
- BR_{BB} - broker to broker relationship.
- BR_{PP} - provider to provider relationship.

Within the generic SDP business model, the SDP may perform the roles of a broker and provider. As a provider, the SDP offers services to consumers and other providers. Hence,



(a) Generic SDP Business Model (b) Decomposed SDP Business Model

Figure 9.2: SDP Business Models

the SDP supports the BR_{CP} and BR_{PP} business relationship points. Also, as a broker the SDP supports interactions between customers and providers. Hence, the SDP supports the BR_{CB} and BR_{BP} business relationship points.

The above business model is generic and may be decomposed to reveal detailed entities and relationships within the SDP business environment. A detailed SDP business model, derived from the generic model, is shown in *Figure 9.2(b)*. The figure illustrates the decomposition of customers. For instance, the consumer is decomposed into a service subscriber and end-user. Numerous providers are also defined, such as connectivity, application, service and content providers. In addition, various brokers, such as service and content brokers, are introduced into the business model.

In *Figure 9.2(b)*, most business relationship points are used between the business entities. For example, the BR_{CB} business relationship point specifies service subscribers may obtain a list of services from service brokers. The BR_{CP} relationship enables the service subscriber to register for selected services from a service provider. In addition, the BR_{CC} relationship allows end-users to use services that service subscribers have registered to use. End-users are allowed to consume services from their appropriate service providers, via the BR_{CP} relationship. Also, the BR_{CP} relationship defines interactions enabling end-users to access their associated connectivity provider. The BR_{PP} business relationship point enables service providers to offer services on behalf of application providers. The various provider interactions are structured using the BR_{PP} relationship. For example, this relationship enables application providers to access and use content provisioned by content providers. The BR_{BP} business relationship point allows service providers or content providers to register their services or content with the appropriate brokers. We also define a BR_{BB} business relationship point between content brokers. This enables content brokers to search for content registered with other content brokers.

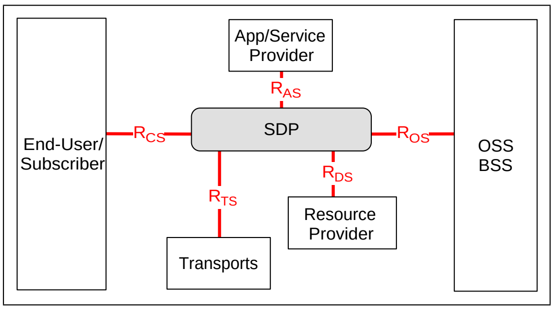


Figure 9.3: SDP Reference Model

As shown in the business models, business relationship points enable convergence of business objectives between diverse business entities. Hence, the relationship points would be defined as rules and policies to ensure converged business objectives are achieved. The SDP and its services provide the required functionality to implement and enforce these business relationship points. However, SDP services must ensure standardised interactions between telco, SDP and its various customers. This standardisation ensures managed and consistent interactions between the SDP and all its customers.

9.1.4 Reference Model

We need a reference model for the SDP that uses reference points to promote standardised interactions between the SDP and its customers. Interactions between the SDP and its customers are specified as business relationship points. These business relationship points prescribe interactions that are allowed and not allowed. In a SDP reference model business relationship points are formalised as reference points. Reference points are implemented as SDP service interfaces. These interfaces being the consumption, client and management interfaces. Also, interfaces are exposed by either building block or composite SDP services. Thus, a SDP reference model formalises the SDP and customer interactions using standardised SDP service interfaces.

The SDP reference model shown in *Figure 9.3* confirms the reference model proposed in *Chapter 2 Figure 2.6*. The reference model is also similar to the SOA derived SDP reference model shown in *Figure 7.5(b)*. The SDP reference model contains all of the entities and reference points used in other derived SDP reference models. These entities include end-users, application providers, content providers, service providers and the telco network.

Figure 9.3 shows the basic entities required by the SDP to standardise its services. These entities fulfil one or more of the consumer, provider or broker roles, defined in the previous SDP business model. Also, business relationship points are now formalised as reference points between the various reference model entities. The transport networks and OSS/BSS entities belong to the telco. Transport networks include legacy and new communication infrastructure and their associated network functions. The OSS/BSS includes management systems and functions that administer the transport infrastructure. These entities provide the SDP with access to telco transport and OSS/BSS functions. The reference model also shows external entities interacting with the SDP via its services. These entities include end-users, service subscribers, application providers, service providers and resource providers. End-users represent individuals that use SDP services to access applications. Subscribers represent external enterprises that act on behalf of many end-users. Application providers use SDP services to create applications that are used by end-users. Application providers also include brokers since they provide applications to end-users, other application providers, service providers and resource providers. Service providers provide services that are used by the SDP. Resource providers use SDP services to contribute data to support application providers. As a result, resource providers may include media distributors, television broadcasters and advertising companies that deliver content to end-users.

The reference model promotes standardised interactions between the SDP and various entities by defining reference points. These reference points are similar to those defined in our SOA derived SDP architecture. These include the horizontal R_{CS} and R_{OS} reference points and the vertical R_{AS}, R_{TS} and R_{DS} reference points. These interactions are realised using SDP service interfaces. Thus, the SDP reference model provides a means to standardise access and usage of services between telco, SDP and its customers.

9.1.5 Overall Management

We require the SDP to incorporate telco OSS/BSS functions such that standard-based SCE and SME are defined. The SDP incorporates the telco OSS/BSS by defining services that abstract its functions. These services can be classified as management services. Management services provide functionality, such as configuring transport networks, adjusting consumer terminal properties and administering customer services. These management services are used via their consumption interfaces by management applications. Management applications orchestrate management service interfaces to implement telco business processes. As an example, a complete set of telco business processes are defined by the eTOM that is discussed in *Chapter 6*. Also, limited management services and interfaces used to implement eTOM are defined by [133].

Management service interfaces contribute to the implementation of the R_{AS} reference point shown in previous reference model. The reference model promotes the standardisation of these management service interfaces. Thus, various internal and external SCE and SME may use these management service interfaces to have consistent access to telco OSS/BSS

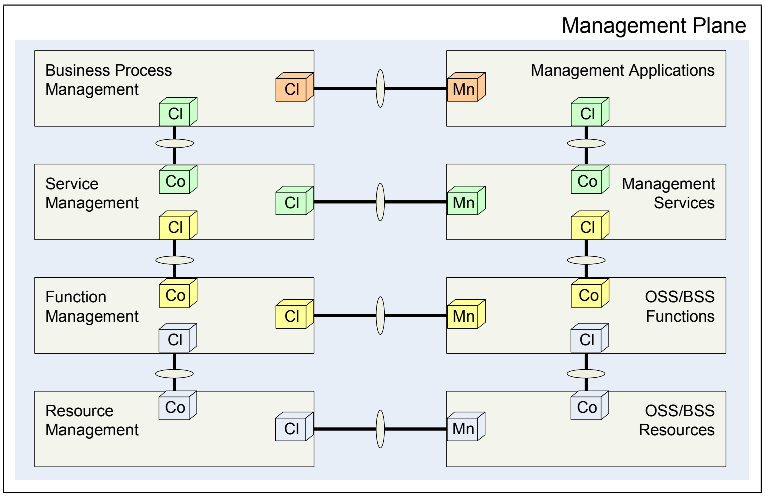


Figure 9.4: SDP Management Architecture with Interfaces

capabilities. We have already derived a SDP management architecture using TMN, TOM and eTOM to illustrate business processes and management services. We extend the derived SDP management architecture to show more detailed layers and the management service interfaces. The extended management architecture is shown in *Figure 9.4*.

The extended SDP management architecture shows a hierarchy containing management applications, management services, OSS/BSS functions and resources. Management applications use their client interfaces (Cl) to access management services via their consumption interfaces (Co). Management applications represent business processes. Management services use their client interfaces to access OSS/BSS functions via their consumption interfaces. OSS/BSS functions use their client interfaces to access OSS/BSS resources via their consumption interfaces. These management applications, services, functions and resources are used to manage telco operations, as well as telco business objectives. However, the extended SDP management architecture shows how the SDP is also managed.

The extended SDP architecture shows interconnected management layers containing administrative services. The business process management layer contains services that administer management applications via their management interface (Mn). The service management layer contains services that administer management services via their management interface. The function management layer contains services that administer OSS/BSS functions via their management interface. The resource management layer contains services that administer OSS/BSS resources via their management interface. The administrative services ensure the SDP satisfies traditional telco network "quality requirements" [69] such as high availability, security, reliability, scalability and fault tolerance.

9.1.6 Architectural Structure

We need an architecture for the SDP that is defined using various design patterns. Unlike the proposed SDP architecture shown in *Figure 1.3*, the new SDP architecture uses technology neutral design patterns to structure the various SDP services. By structuring the services, the new architecture implements SDP reference model reference points as service interfaces. In addition, the new architecture satisfies the SDP business model's business relationship points, by interacting with external IT-using enterprises. Thus, the new SDP architecture supports telco and IT infrastructure interworking, that is, telecom-IT convergence.

We have uncovered technology neutral design patterns from the previous chapters' derived SDP architectures. These design patterns are layers, domains, planes and GSOAs.

Layers:

We use layers as a primary modeling tool to structure the SDP architecture. Layers provide a means to *horizontally* group and structure a collection of related entities, such as SDP services. As a result, layers separate varying levels of functionality. Within layers entities communicate via the client-server or peer-to-peer paradigm. Examples of layers used with telecom and IT-based architectures are numerous. Architectures use layers to model switching levels and service levels [76]. We illustrate layers in a SDP architecture shown in *Figure 9.5*.

In the figure we model switching levels as *Network Resource Layers*. These layers group physical network elements and their network-oriented functions that are abstracted by the SDP and its services. For the SDP, service levels are modeled as *Intelligent Service Layers*. These layers group SDP services and applications. These service layers' services are realised as either building block services (BBS) or composite services (CS).

Service layers use lower resource layers, by accessing their technology-specific functions. Service layers simplify access to these functions by using their technology-neutral service interfaces. Thus, service layers hide the complexities of using lower resource layers. Besides abstracting lower resource layers, service layers simplify each other. For instance, multiple service layers are hierarchically structured with topmost service layers simplifying access to lower service layers. Some service layers are also specific. For example, data stores located in a resource layer may have their functions abstracted by services in a separate intelligent service layer. As a result, this separate service layer may provide only data-centric services.

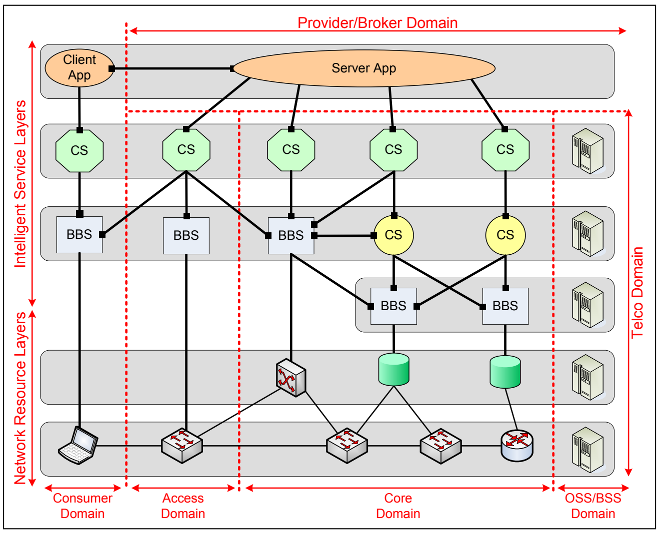


Figure 9.5: Structuring a SDP Architecture using Layers and Domains

Domains:

We use domains as a secondary modeling tool to structure the SDP architecture. Domains are used in many standardised telecom and IT architectures. Domains represent areas of division across the telco network by ownership and functions. Division by ownership relates domains to the various entities that are defined in the SDP business model. As a result, services, applications and functions residing in a domain may belong to a specific business entity. Division by function implies that implemented services, interfaces, applications and functional entities operate across physically distributed equipment and locations.

Domains *vertically* structure the distribution of layers and their content among areas of interest [69]. We illustrate domains in the SDP architecture shown in *Figure 9.5*. In the figure resource and service layers intersect various domains. These domains include customer, provider/broker, telco, access, core and OSS/BSS domains. Services, applications or functions contained within layers communicate synchronously and asynchronously across one or more domains. For instance, some application and services communicate vertically across layers but within the same domain. Also, client and server applications communicate horizontally across domains but within the same layer. Horizontal communication may also occur between network functions contained within a layer.

Similar capabilities, such as services, applications and functions, contained in various layers are grouped into specific domains. For example, all capabilities on the customer device are grouped within the customer domain. Also, an external enterprise that develops applications manages its capabilities within an application provider domain.

Planes:

Specific layers and domains may be grouped together to focus on particular SDP concerns. Focusing on all intelligent service layers and their distribution across domains, centres on creating applications via SDP service interfaces. In addition, these grouping may involve permutations of intelligent service layers and various domains. These groupings are managed with the help of an abstraction tool called planes. Planes are reused in many telecom architectures to abstract complexities, such as lower architectural layers, their contents, implementations and domain distributions.

Based on the previous chapter outcomes, we use a particular type of plane to structure the SDP. This plane provides *middleware* based functionality and is called the middleware plane. The middleware plane provides generic mechanisms to support distributed communication across the SDP, telco and its customers. In addition, the middleware plane supports both horizontal and vertical communication between services, application and functions that contribute to the SDP. Thus, the middleware plane abstracts one or more layers and their distribution across one or more domains.

By providing the middleware plane, complexities associated with distribution are hidden. For instance, the middleware plane enables distributed services to communicate independently of their location. To enable these interactions the middleware plane provides mechanisms to abstract underlying communication and computing infrastructure. This infrastructure may include the telco transport networks and other supporting systems. In addition to abstracting infrastructure, the distribution plane hides implementation details of services and applications. As a result, diverse services may interact with each other and be invoked by diverse application implementations. Thus, the middleware plane hides various complexities of the SDP, so as to simplify relationships and interactions between SDP, services, applications, functions, telco and its customers.

An illustration of planes used in modeling a SDP architecture is shown in *Figure 9.6*. In the figure, planes group layers according to their similarities. This is evident in the service middleware plane abstracting all intelligent service layers. Also, some layers may operate within distinct planes. For instance, layers abstracting service control functionality are grouped into a control plane. The figure also defines a management plane that intersect all layers and planes. Hence, all layers, their contents and planes are managed.

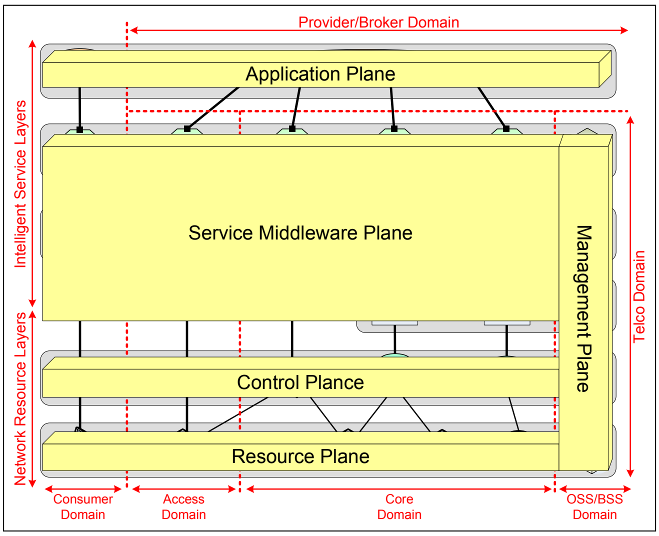


Figure 9.6: Structuring a SDP Architecture using Planes

Generic Service Oriented Architectures:

The GSOA is an outcome of our SOA derived SDP architecture discussed in *Chapter 7*. It encompasses technology neutral concepts that are extracted from the web service SOA and enterprise SOA. We use the GSOA as a technology neutral design pattern to structure service platform architectures. For example, GSOAs structure our IMS derived SDP architecture.

The GSOA represents a container for services and applications. In the GSOA, services define interfaces that abstract access to other services, infrastructure functions or physical systems. The services expose their interfaces to applications. Application orchestrate service interfaces to provide specific functionality. The GSOA provides a distribution plane that services and application operate across. The distribution plane hides various complexities from services and applications. For instance, the distribution of services and applications are hidden from each other. Also, the computing infrastructure supporting service and application communication is abstracted.

GSOAs can be used to elaborate reference points in reference models. As a result, GSOAs contribute towards formalising relationships between reference model entities. Multiple GSOAs can be used to replace layers that are structured according to their level of abstraction. Thus, GSOAs can be layered to mimic abstraction hierarchies. GSOAs abstract the distribution of layers across domains by using it distribution plane. As a result, GSOAs hide distribution complexities if used to structure distributed systems. The GSOA can be used to replace middleware-based planes in service platform architectures. This is possible since the GSOA distribution plane provides middleware capabilities.

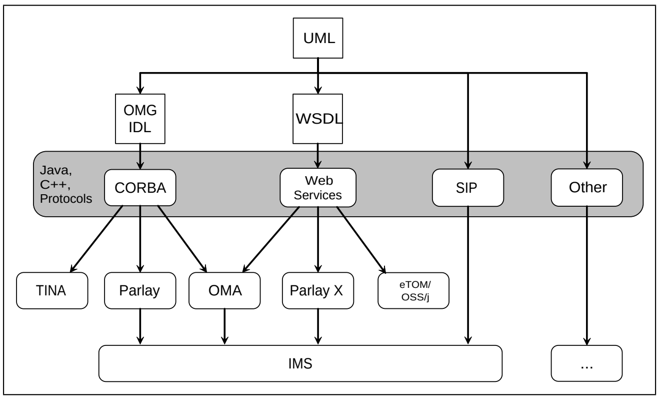


Figure 9.7: Example Technology Map

9.1.7 Standards-based Implementation

We need technologies to implement SDP architectures. In previous chapters we have derived SDP architectures to be void of any technological influences. As a result, these SDP architectures are implementable using a variety of technologies. However, these technologies must satisfy the other SDP requirements and implement SDP services, interfaces, applications, middleware and functions. We motivate the use of open standards-based technologies for implementing the SDP.

By using open standards, SDP implementations remain consistent and *interoperable*. However, vendors may choose to implement the SDP using some proprietary technologies. If so, vendors must implement at least two parts of the SDP using open standards: the topmost *north-facing* service interface and bottommost *south-facing* service interface. The north-facing interface represents the consumption interfaces of services exposed to customers. Basic functionality exposed by these interfaces support application development and resource provisioning. Hence, standardised north-facing interfaces promote portability of applications and content across SDP implementations.

The SDP's bottommost south-facing interface represents the client interface of services that have access to telco network resources and capabilities. These resources and capabilities include telco transport networks, OSS/BSS, databases and media gateways. These interfaces enable SDP services to invoke network functions independently of technologies and distribution. Hence, standardised south-facing interfaces enable portability of SDP implementations across different telco networks. As a result, different vendor SDPs may be consistently used across various telco networks.

A variety of technologies are available to implement the SDP, its service and interfaces. We illustrate an example technology map, in *Figure 9.7*, showing available technology choices

Complexities	Middleware Technologies					
	ASE/ROSE/TCAP	RPC	RMI	CORBA	SOAP	Internet (DNS, DHCP,..)
Distribution	√	√	√	√	√	√
Location	√	√	√	√	√	√
Implementation	√	√	√	√	√	√
Transaction	√			√		
Synchronisation	√	√		√		
Quality of Service				√		
Scalability	√			√		
Security	√			√		
Fault Tolerance				√		

Table 9.1: Examples of Middleware Technologies for SDP Architectures

for the SDP. In the figure, the Unified Modelling Language (UML) [134] is used to describe and model the SDP services independent of technologies. The UML is converted into various types of implementation neutral representations, such as Interface Definition Language (IDL) [82] or WSDL. These representations are implemented using various technologies, such as C++, Java. Some representations are also implemented using complex protocols, such as the SIP.

Many of the above technologies are incorporated into existing *standardised* telecom and IT-based solutions, such as TINA, Parlay and the Open Mobile Alliance (OMA) [50] Service Environment [51]. The existing solutions are also included in newer standards such as the IMS service layer. All these solutions provide standardised services that may be easily incorporated into a SDP implementation. In addition, the above standards specify technologies for their implementations. For instance, diverse middleware technologies are used to implement their services, applications and interfaces. We present a list of middleware technologies used by these standards in *Table 9.1*. The table illustrates middleware technologies and examples of the various complexities they abstract. A blank table entry indicates the middleware does not abstract that complexity.

9.2 Architecture

Based on the SDP business model, reference model, definition and requirements, we derive a SDP architecture. This architecture represents the SDP framework since it is highly generic and extendable. Hence, it provides concepts and the structural foundation for other SDP architectures to be defined. The architecture is defined using service, interface, layer, domain and plane abstractions.

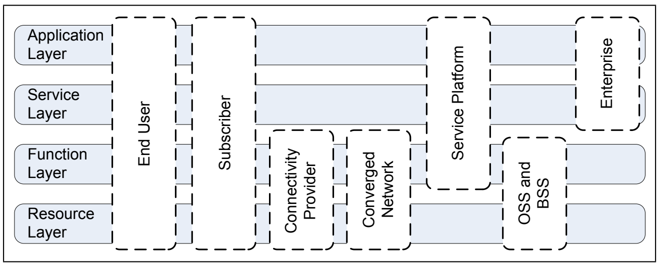


Figure 9.8: SDP Framework Layers and Domains

In *Figure 9.8* we provide a simplified view of the SDP framework using layers and domains. In this model we use four layers to depict the basic functionality contained within the SDP. These four layers are generalisations of layers identified in the previous chapters IN, TINA, Parlay, eTOM, SOA and IMS derived SDP architectures. The application layer contains telco and external enterprise applications that access SDP services. The service layer contains SDP services, with some exposing interfaces to external enterprises. The function layer contains telco network functions that provide the underlying capabilities used by some SDP services. The resource layer contains physical elements that provide a variety of telco network functions.

In addition to layers, the model uses functional domains to distribute the layers and their contents. The domains correspond to the business model entities and reference model entities that benefit from interacting with the SDP. The end-user domain represents the consumer of customer services. The subscriber domain is a special telco or external IT-using enterprise that operates on behalf of the end-user. This enterprise provides the mechanisms to enable end-users to subscribe, locate, consume and pay for customer services. The connectivity provider domain is also a telco or external enterprise that provides the underlying communication mechanisms required by end-users to access their customer services. The converged network domain illustrates an integrated collection of networking resources and capabilities. These networks include telco, IT and Internet networks. The service platform domain represents the heart of the SDP, that is, it contains the collection of distributed services with their interfaces. The OSS/BSS domain represents the telco management platforms that manage the SDP and are reused by SDP services. The enterprise domain represents external IT-using enterprises that use SDP services. The enterprise domain may include application developers, service brokers and media broadcasters.

To provide a fuller representation of the SDP framework we use *Figure 9.9*. The figure shows the SDP exposing numerous service interfaces to entities within its environment. These service interfaces are managed within the appropriate layers. In the figure we decompose the SDP application and service layers, since they represent the core of the SDP. The figure shows the SDP with three application and service layers. These are the simple,

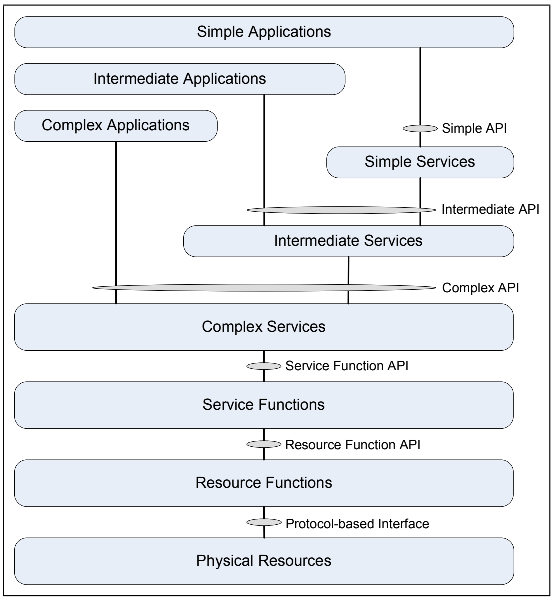


Figure 9.9: SDP and its Environment: Expressing the Full Layers of the SDP Framework

intermediate and complex application and service layers. The simple application layer contains applications that use the simple service layer. These applications do not use the full capabilities of the network since the simple services provide a highly abstracted view on the available network resources and capabilities. The intermediate application layer contains applications that use the intermediate service layer. The intermediate applications use network functionality via the intermediate services, that abstract limited complexities of the network resources. The complex application layer contains rich applications that use the complex service layer. Complex services enable the full capabilities of the underlying network to be invoked by complex applications. Hence, complex services provide limited abstractions of the underlying network resources and capabilities. In addition to these application and service-oriented layers, the SDP framework decomposes the function layer into service function and resource functions layers. The service function layer exposes service-oriented capabilities to complex services, while the resource function layer provides network specific capabilities to implement the higher service functions.

The figure shows each of the service layers exposing consumption interfaces or APIs. These APIs are used by other layers. For instance, simple applications only have access to simple services via the simple API. The intermediate service layer API is used by both intermediate applications and simple services. Simple services use this API to fulfil requests from simple applications. Also, the complex service layer API is used by both complex applications and intermediate services. This complex API is used by intermediate services to fulfil requests from intermediate applications and simple services. Within the service layers internal APIs may also be defined by their services. These internal APIs include client and management interfaces. However, we do not show these internal interfaces in the figure. Lower interfaces

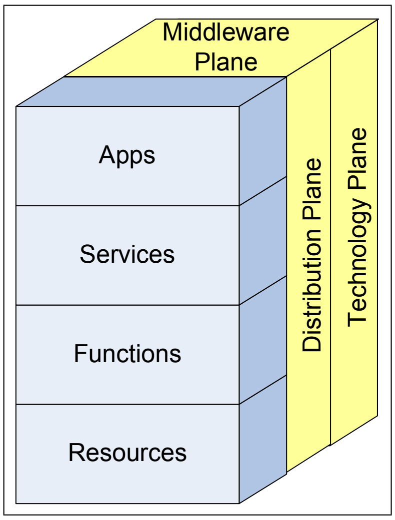


Figure 9.10: SDP Framework Planes

are also shown in *Figure 9.9*. These interfaces include APIs for both service and resource functions. In addition, a protocol-based interface is used to access the underlying converged network resources and capabilities.

Service interfaces promote the standardisation of the SDP by implementing the reference points of the SDP reference model, shown in *Figure 9.3*. However, service interfaces are used between layers, across domains and within layers. Hence, reference points are also implemented across layers and domains. This implies service interfaces are accessed and used across these layers and domains. For example, the horizontal R_{CS} reference point is implemented across the application layers and the end-user and subscriber domains. The horizontal R_{OS} reference point is implemented across the function layers and between the service platform and OSS/BSS domains. The vertical R_{AS} and R_{DS} reference points are implemented across the application, function and resource layers. In addition, these reference points are implemented across the enterprise and converged network domains. The vertical R_{TS} reference point is implemented across the function and resource layers. Also, the connectivity and converged network domains contribute to the implementation of this reference point.

We further provide detail on the SDP framework by adding planes. Planes are used to abstract service distribution across domains and the underlying technologies used to support this distribution. In addition, planes abstract implementations, such that diverse service, applications and functions may communicate via their interfaces. We illustrate the SDP framework planes in *Figure 9.10*. In the figure, we use a single middleware plane to abstract both distribution and technologies within the SDP. This plane may also contain management capabilities, such as services that administer the operation of each layer across the appropriate domains.

9.2.1 Using the GSOA Building block

In the previous chapters we have identified and used the GSOA as a building block for defining service platform architectures. In addition, we have shown the GSOA to be multi-functional design pattern, that is, it elaborates reference points, layers and domains. In addition, it provides abstractions to hide distribution complexities. Hence, to complete the SDP framework we integrate multiple GSOAs into the SDP's structure. The completed SDP framework, using the GSOAs, are shown in *Figure 9.11*.

In the figure, GSOAs are used to manage the various SDP services and their interfaces. Also, the GSOAs manage the diverse range of applications and converged network infrastructure. Each layer of the SDP framework is structure as a GSOA, using the distribution plane as a middleware bus. Hence, we have simple, intermediate and complex middleware that cuts across various domains. In addition, the GSOA and its middleware is used to structure the lower distributed function and resource layers. The figure illustrates the middleware as segmented across the various domains. However, the segmented middleware integrates using horizontal internal interfaces that support all communication across, SDP, IT-using enterprises, end-users and other telcos.

The GSOA's contain the various applications, services, functions and resources. In addition, it manages access to these entities via their interfaces. As shown in *Figure 9.11*, the entities communicate independent of their layer and domain. For example, communication occurs vertically and horizontally across multiple layers and multiple domains. This communication is supported by the various middleware. The GSOAs are also used to manage the layer specific resources, such as service information. This is achieved by integrating information repositories onto the middleware bus and exposing its functionality as distribution, technology and implementation neutral service interfaces.

The SDP framework benefits from using the GSOA since it structures layers and abstracts the functional distribution of these layers. In addition, it provides a manageable container for layer specific applications, services, functions and resources. Also, like layers the GSOAs abstract lower GSOAs. The SDP framework inherits GSOA properties, that is, it is technology, implementation and distribution neutral. As a result, the SDP framework may be implemented using various technologies that are standard-based.

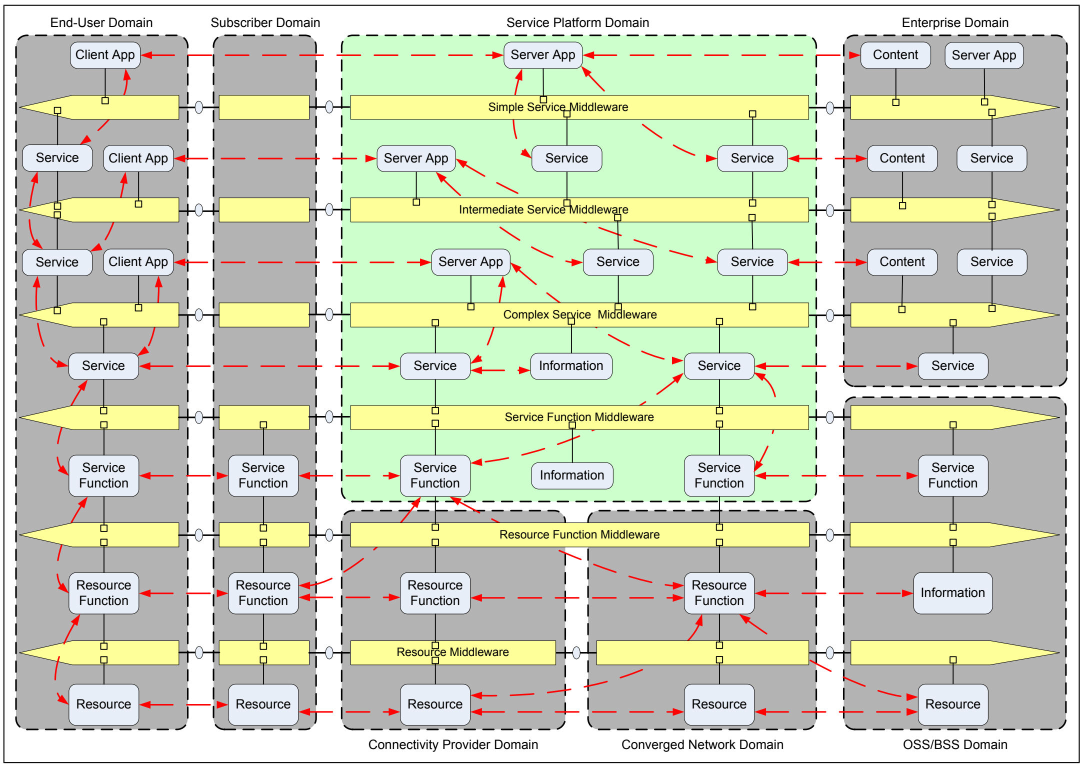


Figure 9.11: Complete SDP Framework

9.3 Results

The SDP framework represents the visualisation of a collection of SDP concepts, principles and abstractions. The various concepts include a business model to encompass the SDP's business objectives, that is, creation, delivery and management of customer services by various external IT-using enterprises. A corresponding reference model is also used to ensure that all external access to the SDP is formalised. Also, the reference model ensures SDP's access to converged network resources and capabilities are also formalised. The SDP framework, as presented in *Figure 9.11*, satisfies SDP requirements and is based on the SDP business and reference models. In addition, the framework is highly generic and may be extended, reduced or decomposed. However, any framework modification must adhere to the SDP concepts and principles.

The SDP framework benefits from using GSOAs. The GSOAs ensure distribution, technology and implementation independence that is inherited by the framework. The GSOAs are used across all layers and domains of the framework. As a result, the SDP framework presents a consistent structure using GSOA abstractions, such as services, interfaces and middleware. The GSOA middleware planes used in the SDP framework provide essential mechanisms to manage the various SDP services, applications, functions and resources. In addition, the middleware mechanisms may provide additional functionality that contribute to the SDP framework.

The SDP framework is generic. To illustrate this property we map existing standards onto the various layers, domains and GSOAs. We use *Table 9.2* to illustrate an example mapping of the IMS onto the SDP framework. In the table empty slots indicate the IMS does not provide abstractions or standardised components to satisfy the SDP framework. Additional mappings of both standardised and proprietary SDP architectures onto our SDP framework are illustrated in *Appendix A*.

9.4 Summary

In this chapter we glued together various concepts, principles and abstractions from previous chapters into the SDP and its framework. As a result, we defined the SDP and provided a list of requirements. All requirements are void of any biases, such as technology or vendor. These requirements included a business model to motivate the business case for the SDP. The business model also showed the endless possibilities for business relationships between SDP, telco and diverse external IT-using enterprises. We illustrated a generic reference model that promotes standardised communication between the business entities. The standardisation across the reference model is achieved using the SDP services' interfaces. We have shown SDP services offering client, consumption and management interfaces. Also, we have classified SDP service into building block and composite categories. To visualise the various concepts and principles uncovered in the SDP requirements, we defined the SDP framework. The framework consists of multiple application and service layers. These layers form the core of the SDP. We showed the various forms of interfaces between the application and service layers. These interfaces provide standardised communication required within the SDP reference model. We used domains to distribute the SDP framework layers and illustrate the distributed nature of the SDP. We incorporated a distribution plane to manage the complexities associated with distribution. To manage the collection of abstractions within the SDP framework we used the GSOA. The GSOAs provided the container for layers, their applications, services and interfaces. In addition, the GSOAs provided the means to manage distribution using middleware abstractions. We also showed the SDP framework inheriting the distribution, technology and implementation neutral properties of the GSOA. Thus, by reusing concepts, principles and abstractions from a wide range of standard-based technologies we defined the SDP framework to promote SDP standardisation.

<table>
<tr><th colspan="8">Generic Service Delivery Platform Architecture</th></tr>
<tr><th>↓ $Layers/Domains$ →</th><th>End User</th><th>Subscriber</th><th>Connectivity Provider</th><th>Converged Network</th><th>Service Platform</th><th>OSS/BSS</th><th>Enterprise</th></tr>
<tr><td>Simple Apps</td><td></td><td></td><td></td><td></td><td></td><td></td><td>Parlay X Apps</td></tr>
<tr><td>Simple Services</td><td></td><td></td><td></td><td></td><td colspan="2">Parlay X Web Services</td><td></td></tr>
<tr><td>Simple Service Middleware</td><td></td><td></td><td></td><td></td><td colspan="3">SOA and ESB-based</td></tr>
<tr><td>Intermediate Apps</td><td></td><td></td><td></td><td></td><td>Parlay App</td><td></td><td>Parlay App</td></tr>
<tr><td>Intermediate Services</td><td></td><td></td><td></td><td></td><td>SCFs</td><td>SCFs</td><td></td></tr>
<tr><td>Intermediate Service Middleware</td><td></td><td></td><td></td><td></td><td colspan="3">RMI, CORBA, SOA and ESB based</td></tr>
<tr><td>Complex Apps</td><td>SIP UE</td><td></td><td></td><td></td><td>gsmSCF and SIP AS</td><td></td><td></td></tr>
<tr><td>Complex Services</td><td></td><td></td><td></td><td></td><td>SCS, IMSSF,
SCIM and SIP AS</td><td></td><td></td></tr>
<tr><td>Complex Service Middleware</td><td>SIP</td><td></td><td></td><td></td><td>SIP and Diameter</td><td></td><td></td></tr>
<tr><td>Service Functions</td><td></td><td></td><td></td><td></td><td>CSCFs, SLF and HSS</td><td></td><td></td></tr>
<tr><td>Service Function Middleware</td><td></td><td></td><td></td><td></td><td>SIP and Diameter</td><td></td><td></td></tr>
<tr><td>Resource Functions</td><td></td><td></td><td>P-CSCF and PDF</td><td>MGCF and BGCF</td><td></td><td>SCF, ECF, CGF, ...</td><td></td></tr>
<tr><td>Resource Function Middleware</td><td></td><td></td><td>SIP</td><td>SIP</td><td></td><td>Diameter</td><td></td></tr>
<tr><td>Resources</td><td></td><td></td><td>GGSN, SGSN and RAN</td><td>SGW and MGW</td><td></td><td></td><td></td></tr>
<tr><td>Resource Middleware</td><td></td><td></td><td colspan="2">Protocols</td><td></td><td></td><td></td></tr>
</table>

Table 9.2: Mapping IMS onto the SDP Framework

Chapter 10

Proving the SDP Framework

To prove the concepts of the SDP framework we define an application that operates across the SDP and provides a service to end-users. The application is created by an application provider who is an IT-using enterprise. The application uses content provisioned by a media provider. The application is enhanced by a service provider to provision a service to end-users. As a result, the service is used by end-users, who obtain access via their service provider. Supporting the application provider, service provider, and end-users are the SDP and its services. In addition, the telco supports all entities by providing transport connectivity and access to converged network capabilities.

We propose to implement a content delivering application that combines voice, video, presence and messaging services into one advanced service. We name this service *Interactive Personalised Tele-Vision (IPTV)*. The service aims to deliver real-time or on-demand television content to end-users. In addition, IPTV enables end-users to use presence related telephony and messaging while viewing their content.

To implement both SDP and IPTV service, we use the following approach. First, we describe the IPTV service requirements. Second, we define the SDP business and reference models that support the IPTV service. Third, we determine services and interfaces required to fulfil business relationships, reference points and IPTV functions. These services are mapped onto the SDP framework GSOAs. Fourth, rather than rebuilding these services and interfaces we reuse existing standard-based technologies to implement the SDP framework GSOAs. Fifth, we detail interactions across the framework involving the IPTV service and SDP services. Last, we show details of the implemented SDP and IPTV service on a physical network.

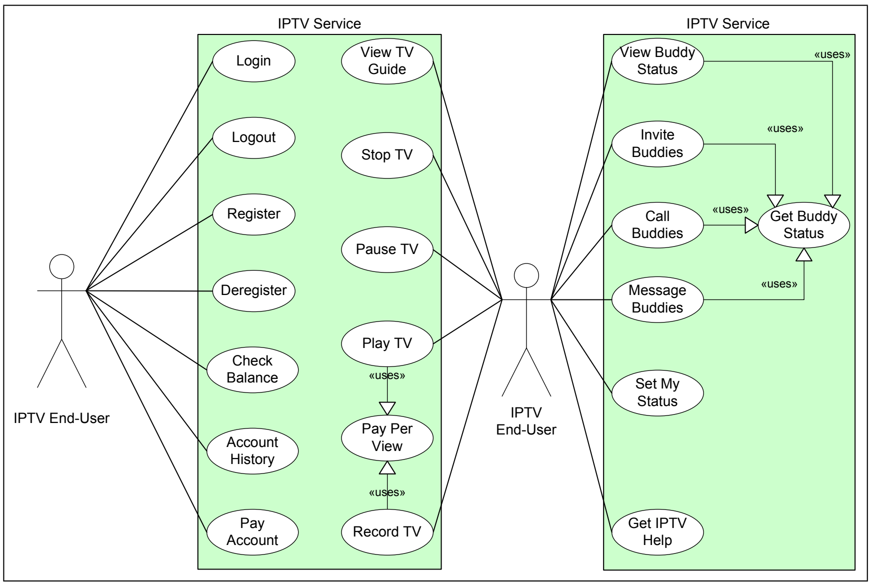


Figure 10.1: IPTV Use Cases

10.1 IPTV Service Description

The IPTV service is described from the perspective of the end-users. Hence, we define use cases describing interactions between end-users and the IPTV service. The IPTV use cases are shown in *Figure 10.1*. These use cases and their descriptions are:

Register: the end-user requests access to the IPTV service. End-users provide their details, such as name, address and bank account details.

Deregister: registered end-users do not require access to the IPTV service. Also, end-users are not billed for the IPTV service anymore.

Login: end-users access the IPTV service by providing usernames and passwords. Once authenticated they are allowed to access the IPTV service.

Logout: end-users request deactivation of the IPTV service, however, they remain subscribed to the service.

Check Account: an end-user requests their account details, such as account details or balance.

Account History: the end-user requests their entire account history.

Pay Account: end-users submit payment towards their IPTV account using various means, such as inputting voucher numbers or credit card details.

Pay Per View: an end-user's account is charged when viewing or recording special types of television content.

View Television Guide: end-users request a list of available television content.

Play, Stop and Pause: end-users control the delivery of television content to their device. The end-user may start, pause or stop television content. The end-user may also be charged for requesting special content. Hence, we include the pay per view use case.

Record: end-users request the service to store live television content to view at a later stage. This scenario may involve billing the end-user. Hence, we include the pay per view use case.

Get Buddy Status: the service obtains presence information for one or more of the end-user's friends. These friends are also registered to use the IPTV service. Examples of presence status includes busy, online, away and do not disturb.

View Buddy Status: the end-user requests the status of one or more friends.

Invite Buddy: end-users invite one or more friends to watch television. Only friends that have an available presence status may join.

Call Buddy Use: end-users initiate a call between themselves and their friends. These calls may be normal voice calls or advanced multimedia calls. Only friends with an available presence status may join the call.

Message Buddy: end-users send messages to their available friends.

Set My Status: end-users set their presence status.

Get IPTV Help: at any time the end-user may require help using the service. Hence, end-users activate an interactive tutorial and follow its voice prompt to troubleshoot service related problems.

10.2 Business Model

To show business entities involved in providing the IPTV service, we illustrate a specific SDP business model in *Figure 10.2*. This business model is derived from the generic SDP business model, shown in *Figure 9.2(a)*.

The figure shows all SDP customers, that is, consumers, providers and brokers involved in providing the IPTV service. Customers include an IPTV application provider, who creates the application to deliver content from a content source. The application provider uses a content broker to locate diverse content sources. The content sources are managed by media providers. The IPTV application is enhanced by a service provider, to enable consumer

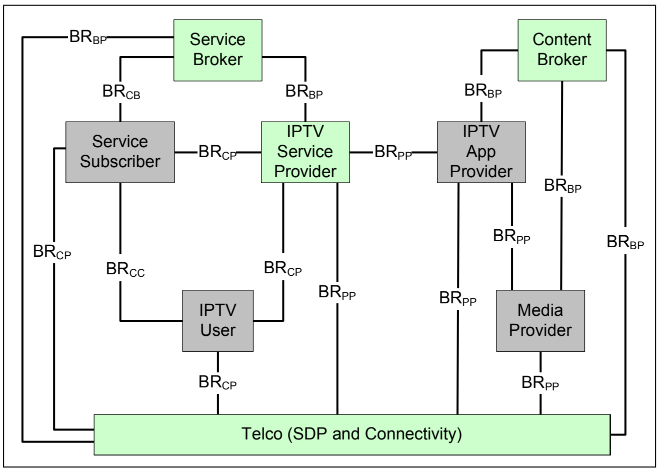


Figure 10.2: SDP Business Model Supporting IPTV

access to an IPTV service. The service provider provisions the IPTV service to service brokers. Service brokers enable consumers to locate the IPTV service. These consumers include individual end-users or service subscribers, who act on behalf of many end-users.

The SDP business model uses all business relationship points to specify business objectives shared between the various customers. For the SDP and IPTV service, the business relationship points specify policies for the business entity interactions. Examples of these policies include:

- IPTV application providers are allowed to access content provisioned by multiple media providers.
- Media providers may register their content with multiple content brokers.
- IPTV end-users are allowed to subscribe for the IPTV service from service providers, but not application providers.
- Service brokers are allowed to offer the IPTV service to both service subscribers and end-users. However, service subscribers may not re-offer the service to other service subscribers.
- The connectivity provider must provide the required QoS to stream content to the IPTV end-user. If the appropriate QoS cannot be guaranteed the IPTV end-user must be notified and provided with alternative means of streaming content.

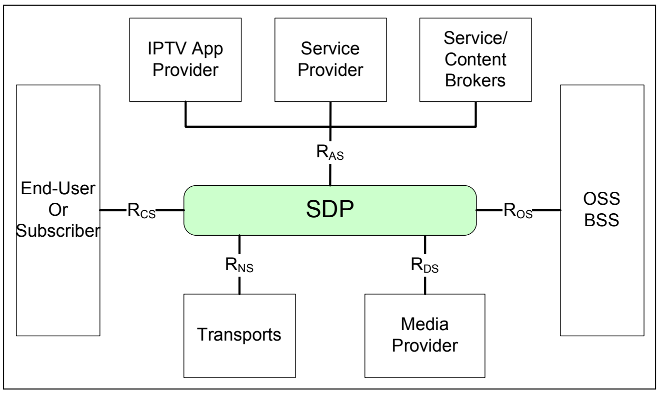


Figure 10.3: SDP Reference Model for IPTV

10.3 Formalising Interactions

The SDP must support the various business relationships to deliver the IPTV service to end-users. These relationships are implemented as numerous interactions between business entities. Also, interactions are formalised to ensure consistency and interoperability between the various consumers, providers and brokers. To determine these interactions we derive a SDP reference model. The reference model is based on the generic SDP reference model shown in *Figure 9.3*. The derived SDP reference model is shown in *Figure 10.3*.

The reference model only shows reference points being used by the SDP to support the IPTV service. In the reference model, the R_{AS} reference point is crucial for supporting IPTV the application provider, service provider and brokers. Also, the R_{DS} reference point is required to ensure management and delivery of content by the media provider. The R_{OS} reference point is also required to enable IPTV end-user access to their billing details and to be billed for using the service. The R_{NS} reference point is required, such that the SDP uses converged network capabilities to deliver the IPTV service and content to end-users. The R_{CS} reference point enables the SDP to communicate with the IPTV end-user, if required.

To specify these interactions we formalise them as interfaces that are exposed by SDP services. In addition, services implementing interfaces contribute logic to satisfy the business relationship policies. To determine these interfaces and associated services we use the SDP framework.

10.4 Services, Interfaces and SDP Framework

For the IPTV service, we use the SDP as the IPTV service provider, service broker and content broker. Hence, SDP service interfaces are required to support these roles and satisfy the associated business relationship policies. However, the SDP must also provide diverse service interfaces for the external IPTV application provider and media provider. These service interfaces aid in IPTV application development and content management respectively.

To uncover the required SDP services and interfaces, we define the use cases shown in *Figure 10.4*. The use cases illustrate functions required by the SDP to perform service provider, service broker and content broker roles. In addition, the use cases show functions provided by the SDP to satisfy external application and media providers. The use cases and their descriptions are:

Set User Presence: the IPTV application changes end-user presence information stored by the SDP.

Get User Presence: the IPTV application requests end-user presence information from the SDP.

Setup Call: the IPTV application requests a normal or multiparty voice/video call to be setup between end-users.

Send Message: the IPTV application requests a message be sent to an end-user.

Find Content: the IPTV application requests a list of specific content from content broker, who is the SDP. The application may query details on content contained in the list.

Deliver Content: the IPTV application requests the delivery of content from a media source to end-users. The content includes audio, video and data.

Register Content: the media provider application registers itself as a content provider with the SDP (content broker). In addition, the application provides information about its available content, such as location, format, description, time of availability and length.

Manage Content: the media provider application updates its available content information that has been registered with the SDP. The media provider application may also remove content from its registration or add new content.

Register Services: the IPTV service provider, who is also the SDP, uses an application to register its IPTV service, such that end-users can locate, register and use it.

Manage Subscribers: the IPTV service provider application requests registration of end-users to access and use the IPTV service. The application may also deregister end-users and verify login information.

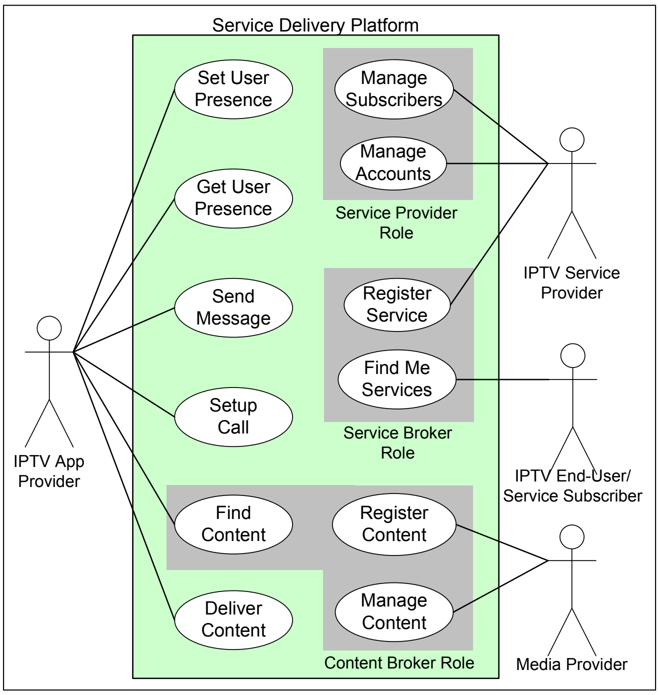


Figure 10.4: SDP Specific Use Cases

Manage Accounts: the IPTV service provider application obtains information about end-users billable accounts. The application may also modify these accounts, such as billing end-users for delivery of special content.

Find Me Services: the end-user application requests the SDP, who is also a service broker, to obtain a list of available services, such as the IPTV service. The end-user application also obtains details on the available services in the list.

Based on these use cases and the previous SDP reference model, business model and IPTV use cases, we define the SDP to provide the following services:

presence service: provides an interface that manages end-users presence information.

call service: provides and interface that enables application to invoke classical and enhanced telephony services.

messaging service: enables messaging functions to be performed on its interface.

stream service: enables applications to invoke its interface, so as to delivery content from a media source to end-users.

content management service: exposes an interface for applications to register, manage and locate content.

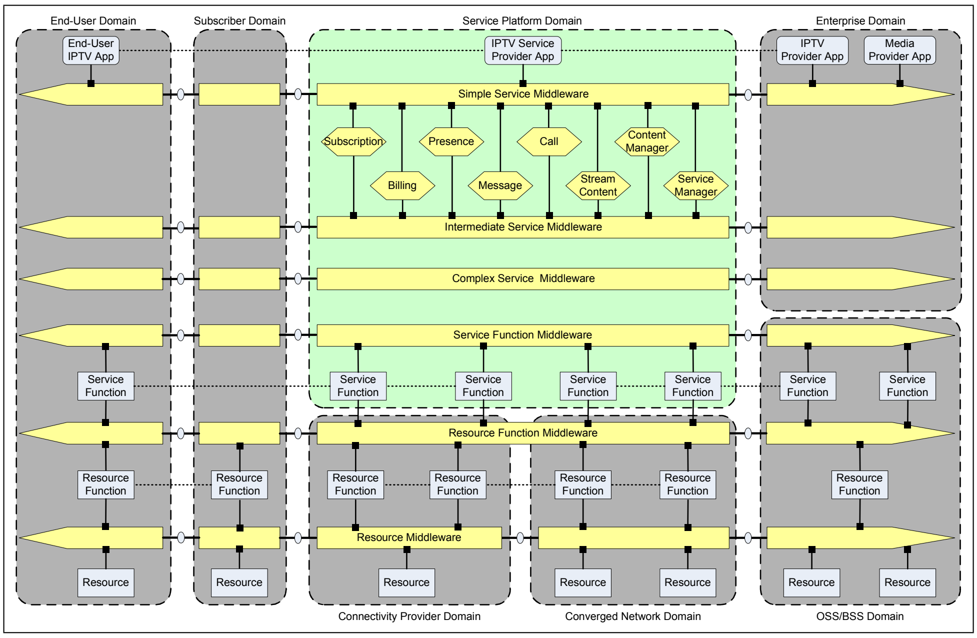


Figure 10.5: SDP Framework with Simple Services Enabling IPTV

service management service: provides an interface to enable end-user applications to locate the IPTV service offered by a service provider.

subscription service: provides an interface for the IPTV service provider application to register, deregister and verify login details of end-users.

billing service: its interface abstracts access to telco OSS/BSS functionality, such as charging end-users and providing account management.

We present the identified SDP services and their interfaces within the SDP framework. This SDP framework is shown in *Figure 10.5*. In the figure, we contain these services in the simple GSOA. This enables IPTV and media provider applications to use converged network capabilities at a high level of abstraction. As a result, the providers do not require extensive telco network knowledge.

To complete the framework we require intermediate and complex services. Also, lower layer functions must be present with their associated physical resources. However, rather than building all GSOAs and physical resources, we reuse existing technologies. These technologies must provide open standard-based service interfaces that are applied across all GSOA layers of the SDP framework.

10.5 Mapping Standard-based Technologies

To implement the SDP framework and provide the IPTV service, we use the standardised IMS and Parlay set of technologies. We use both Parlay and Parlay X APIs to provide service interfaces for the service layer GSOAs. The IMS functional entities provide lower layer abstractions of existing physical network resources. These network resources being the Gateway GPRS Support Nodes (GGSN), media gateways, signaling gateways, media stores and existing telco billing systems. The SDP framework with mappings of Parlay, Parlay X and IMS are shown in *Figure 10.6*.

The IMS functional entities are used to implement both resource function GSOA and service function GSOA. For the resource functions the Policy Decision Function (PDF), Media Gateway Controller Function (MGCF), Breakout Gateway Function and Media Resource Function Processor (MRFP) are used. In the OSS/BSS domain, the IMS provides the Charging Gateway Function (CGF). For the service functions the IMS provides the proxy, interrogating and serving Call Session Control Functions (CSCFs) and Home Subscriber Server (HSS). The IMS also provides end-user, media processing and OSS/BSS service functions, such as the SIP user equipment (UE), Media Resource Function Controller (MRFC) and Charging/Event Collection Function (CCF/ECF) respectively. The IMS service functions invoke resource functions using protocols, such as SIP and Diameter [42]. These protocols hide limited distribution complexities. Hence, the IMS protocols provide implementations of the resource and service function GSOA middleware planes.

The IMS also contributes SIP application servers to the realisation of the complex GSOA. However, we use the Parlay SCS to encapsulate these SIP application servers. Thus, the Parlay SCS implements the complex GSOA. The intermediate GSOA is implemented using the Parlay SCFs. The SCFs represent services that expose their capabilities using APIs. Both SCS and SCFs are implemented using CORBA-based middleware. CORBA abstracts numerous distribution complexities using standard-based middleware services. Hence, it provides a rich implementation for the intermediate and complex GSOA middleware planes.

To abstract the Parlay-based intermediate GSOA, the Parlay X APIs are used. These APIs are implemented as web services. Hence, the simple GSOA is implemented using web service technologies. By using web services, the simple GSOA's middleware plane is implemented as an ESB. The ESB abstracts distribution complexities associated with the web services. Using the web services across the ESB are various web-based applications, such as the IPTV application and media provider application.

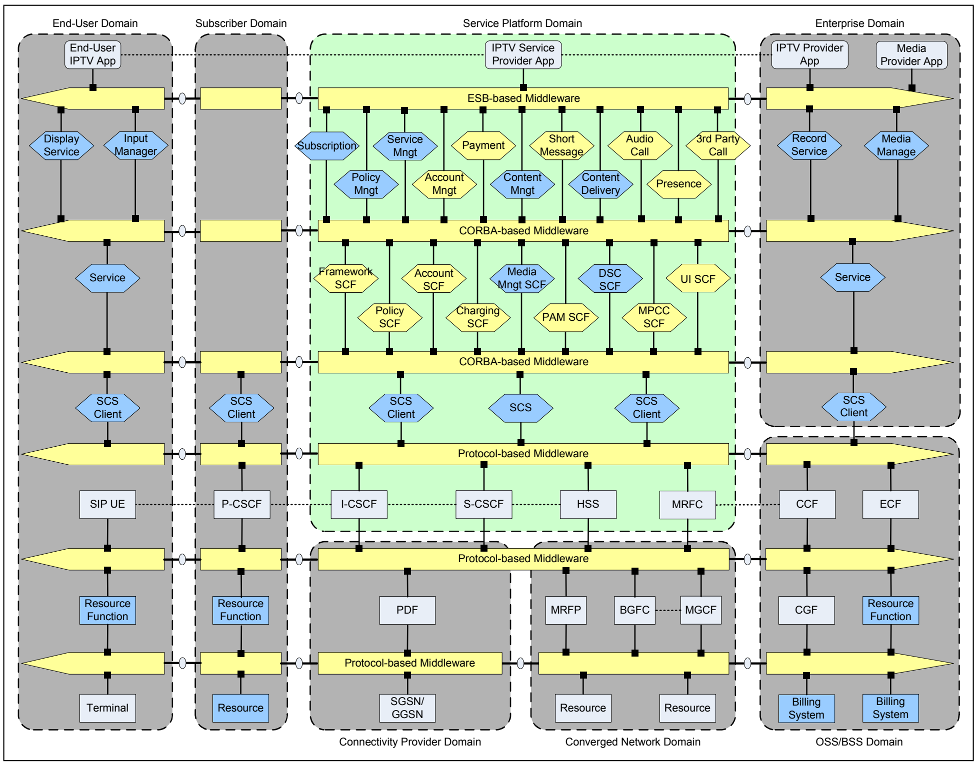


Figure 10.6: Using Parlay and IMS Standards to implement a SDP

10.5.1 Alternatives

In Parlay X, web services may be used to directly access network resources and capabilities [135]. Thus, as an alternative the SDP framework may use Parlay X to access service functions rather than SCFs. In this alternative the intermediate and complex GSOA layers are collapsed into the simple service GSOA. Also, we may completely remove Parlay X as the simple GSOA layer. As a substitute we may use a Parlay-based intermediate GSOA layer to promote application creation and exposure of network abstractions to external enterprises.

Similarly to manipulating GSOA layers, we may alter SDP framework domains. For instance, the service subscriber domain may be removed if the end-user is within his/her home IMS network. Hence, the P-CSCF and resources are removed and not accessed from the customer domain. Also, a telco may assume the access provider role for end-users. In addition, the telco may provide the OSS/BSS capabilities to bill the end-users for accessing and consuming services. The telco may also provide IMS functions and network resources that contributes to the converged network domain. As a result, the access provider, converged network and OSS/BSS domains may merge into a single telco domain.

Mapping of other standardised technologies to the SDP framework may require manipulation of GSOA layers and distributed domains. As a result, specific SDP architectures are created based on these manipulations. Though incorporating technologies, these derived architectures inherit the generic concepts, principles and abstractions of the SDP framework.

10.6 SDP and IPTV Service Implementation

As shown in *Figure 10.6*, we identify existing Parlay X web services and Parlay SCFs that support the development of the IPTV service. We also provide recommendations for new web services, SCFs and SCS logic to support the IPTV service.

A collection of Parlay X web services are identified to provide the IPTV functions. These web services also abstract the necessary Parlay SCFs. These web services are the:

- payment [136] web service that simplifies the charging [137] SCF.
- account management [138] web service that simplifies the account management [139] SCF.
- short message [140] web service that simplifies the user interaction [141] SCF.
- audio call [100] web service that also simplifies the user interaction SCF.
- 3rd party call [99] web service that simplifies the multi-party call control [142] SCF;
- presence [143] web service that simplifies the presence and availability SCF [144].

No suitable web services are defined to provide subscription management, service management, content management and content delivery functions needed by the SDP and IPTV service. However, Parlay defines a framework SCF [90] that provides some subscription and service management functions. Also, Parlay uses a data session control SCF [145] to manage data sessions, that may be modified to stream content. Hence, we identify new subscription management and service management web services that abstract the framework SCF. We also define a content delivery web service to abstract the modified data session control SCF.

Parlay defines a policy management SCF [146] that is used by the telco or external enterprise to define and enforce business rules and contracts. Policies may be defined to regulate SCF usage. Hence, policies control content delivery, locating of services/content, making calls, sending messages and service billing. Hence, we motivate the use of the policy SCF to manage business relationships between the SDP, IPTV application provider and media provider. However, no Parlay X web service is defined to abstract the policy management SCF. This policy management web service enables application and media providers to define their own policies or view telco defined policies. As a result, we identify the policy management web service to be added to the SDP, to fulfil its requirements and IPTV functions.

ESB middleware is used to abstract the distribution of the Parlay X web services. However, ESB solutions include both standardised and proprietary technologies. Hence, we only incorporate standard-based technologies that are used in ESBs, to hide web service distribution complexities. These included basic web services technologies and protocols, such as XML, SOAP and HTTP. As a result, the Parlay X web services use a limited ESB that does not provide additional middleware functionality.

The Parlay SCS, implementing the complex GSOA, enables conversion between Parlay SCF invocations and protocols used to communicate with the IMS functional entities. However, there is a lack of standardisation on the SCS API and the conversion of most SCF invocations to SIP or Diameter protocols. As a result, we add new methods to the limited SCS API and provide a proprietary implementation. In addition, we create a CORBA-based network simulator that hides lower IMS functions and facilitates communication with SCS on its API. The simulator contains distributed SCS clients. These clients simulate network events and communicate with the Parlay SCS across the various SDP framework domains.

10.6.1 Interactions via APIs

The interactions between the standard-based web services, SCFs and SCS APIs follow those defined in the Parlay X, Parlay and IMS standards. Examples of these interactions are shown in *Appendix B*.

Defining the subscription and policy management web service APIs, by abstracting the framework SCF and policy SCF APIs, is a complex task. As a result, we provide recommendations and example sequence diagrams showing these web services and SCFs in *Appendix B*. In the appendix we also provide recommendations for the content management web service API and associated media management SCF API. In addition, we provide example sequence diagrams showing their interactions. In *Appendix B*, we provide recommendations and sequence diagrams for an extended SCS that manages SIP conversions required to satisfy most SCF invocations.

In this section we show, as an example, interactions between the new content delivery web service and modified data session control SCF. We also show interactions between SCS and SCS clients that simulate data session network events.

In *Figure 10.7* we show the end-user initiating the delivery of television content to his/her terminal. The interactions between the applications, web service and SCF using their APIs are as follows:

1. The end-user requests to watch a television programme from the IPTV service provider.
2. The service provider requests the IPTV application provider to start the television programme on the end-user device.
3. The IPTV application provider invokes the newly defined deliver content web service to start a data session between the end-user and media source.
4. The web service creates a data session callback object.
5. The web service invokes the data session control SCF using this new method, to create a abstract representation of the data session.
6. The SCF creates the data session object.
7. The web service sets the charge plan for this session, based on the policies defined for accessing and delivering the required content.
8. The web service requires notification of any changes to the data session.
9. The web service requests the connection of the end-user (client) to the data session.
10. The data session object requests the SCS to create the physical data session in the converged network and connect the end-user. The web service is informed that the client is being connected.
11. The web service requests the connection of the media source (server) to the data session.
12. The data session object requests the SCS to connect the media source to the physical data session. The web service is informed that the server is being connected.
13. Once both client and server are connected, the SCS informs the data session object.
14. The data session object informs the web service callback object that a successful data session is created.
15. The callback object informs the web service that a television programme is being delivered to the end-user.

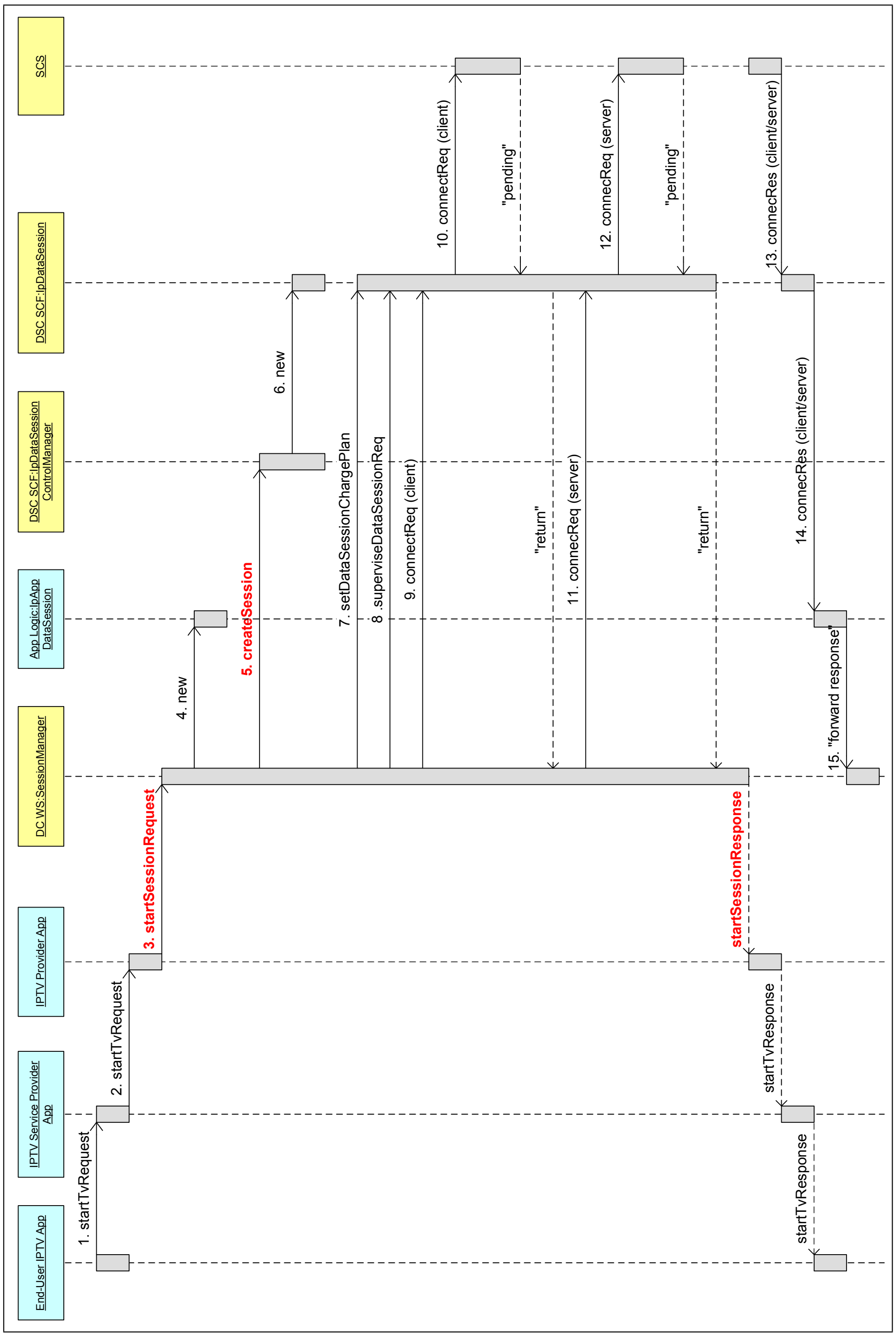


Figure 10.7: Starting a Data Session for Streaming Content

In *Figure 10.8* we show the end-user pausing the delivery of television content to his/her terminal. The interactions between the applications, web service and SCF using their APIs are as follows:

1. The end-user requests the IPTV service provider to pause the current television programme.

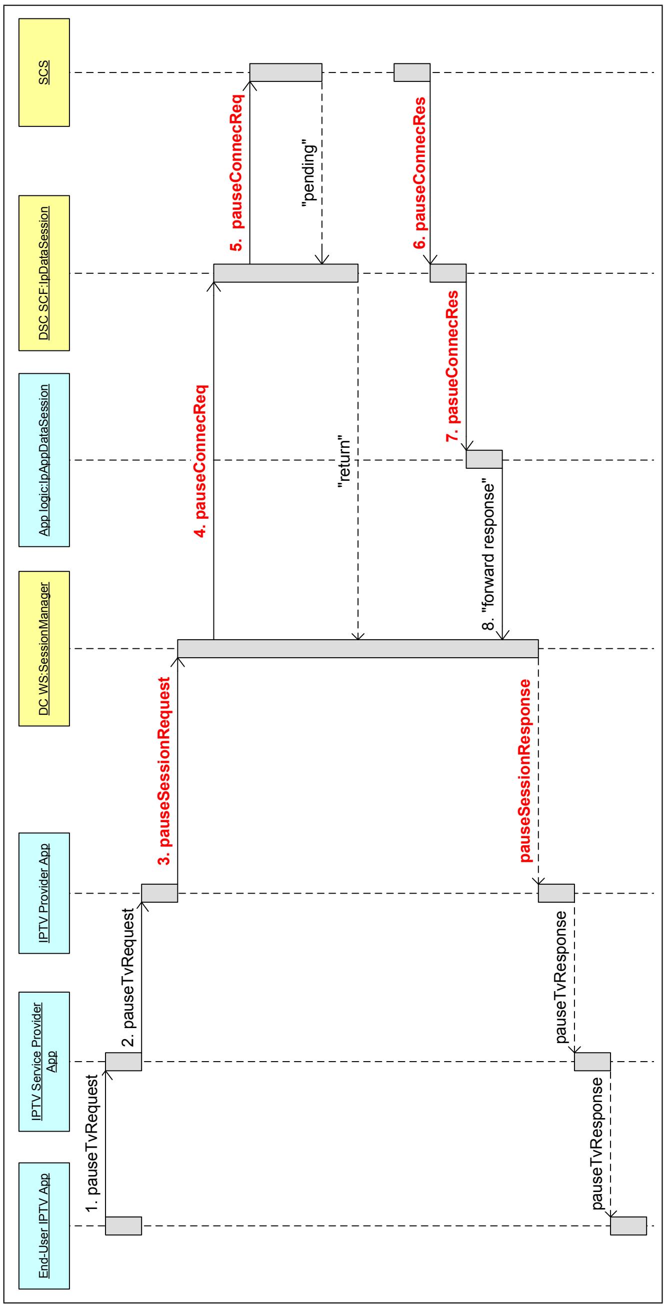


Figure 10.8: Pausing a Data Session Streaming Content

2. The service provider requests the IPTV application provider to pause the current television programme being delivered to the end-user device.

3. The IPTV application provider invokes the newly defined deliver content web service to pause the data session between the end-user and media source.

4. The web service invokes the data session object using this new method, to pause the data session.

5. The data session object requests the SCS to pause the data session in the network.

6. Once the data session is paused, the SCS informs the data session object.
7. The data session object informs the web service on its callback object.
8. The callback object informs the web service of the successful pause request. The response is forwarded to the various applications and end-user.

In *Figure 10.9* we show the end-user resuming and stopping the delivery of television content to his/her terminal. The interactions between the applications, web service and SCF using their APIs are as follows:

1. The end-user requests the IPTV service provider to resume the paused television programme.
2. The service provider requests the IPTV application provider to resume the currently paused television programme being delivered to the end-user device.
3. The IPTV application provider invokes the newly defined deliver content web service to resume the data session between the end-user and media source.
4. The web service invokes the data session object using this new method, to resume the paused data session.
5. The data session object requests the SCS to resume the data session in the network.
6. Once the data session is resumed, the SCS informs the data session object.
7. The data session object informs the web service on its callback object.
8. The callback object informs the web service of the successful resume request. The response is forwarded to the various applications and end-user.
9. The end-user requests the IPTV service provider to stop the current television programme.
10. The service provider requests the IPTV application provider to stop the current television programme being delivered to the end-user device.
11. The IPTV application provider invokes the deliver content web service to stop the data session between the end-user and media source.
12. The web service invokes the data session object using its existing release method, to stop the data session. The web service may also delete its callback object, since it is no longer being used.

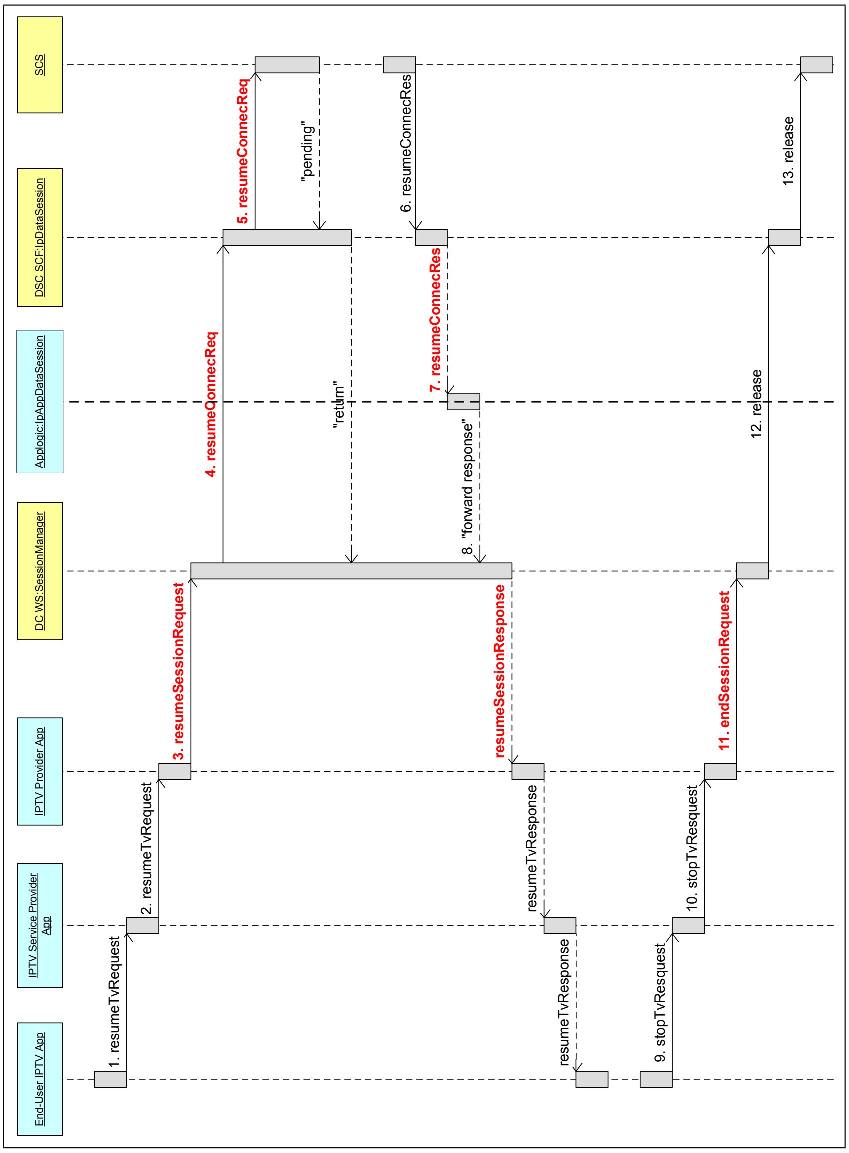


Figure 10.9: Resuming and Stopping a Data Session Streaming Content

13. The data session object requests the SCS to stop the data session in the network and release all used resources.

In *Figure 10.10* we show the SCS and data session SCS client interactions. The data session SCS client forms part of our network simulator that abstracts lower network events and interactions. The interactions between SCS and data session SCS client APIs are as follows:

1. The SCS requests the simulator to create a data session and attached a client to the session.

2. The simulator simulates the session setup and the attachment of a client using its address.

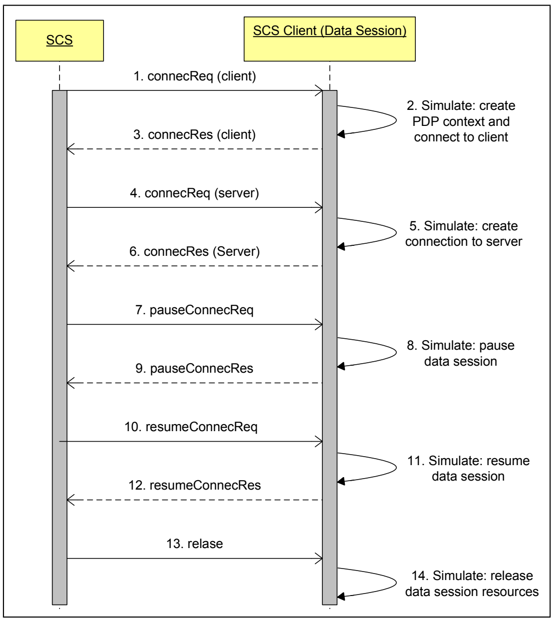


Figure 10.10: Simulating Network Data Session Manipulation

3. The simulator informs the SCS of the successful creation of a data session and attached client.

4. The SCS requests the simulator to attach a server to an existing data session.

5. The simulator simulates the attachment of the server using its address.

6. The simulator informs the SCS of the successful attachment of a server.

7. The SCS requests the simulator to pause an existing data session, such that both client and server cannot communicate across the session.

8. The simulator simulates the session being paused.

9. The simulator informs the SCS that the data session is paused.

10. The SCS requests the simulator to resume and existing data session that has been paused.

11. The simulator simulates the resuming of a session, such that client and server may communicate.

12. The simulator informs the SCS that the data session has been resumed.

13. The SCS requests the simulator to stop and release an existing data session.

14. The simulator simulates the disconnection of the client and server from the data session and the release of network resources used by the data session.

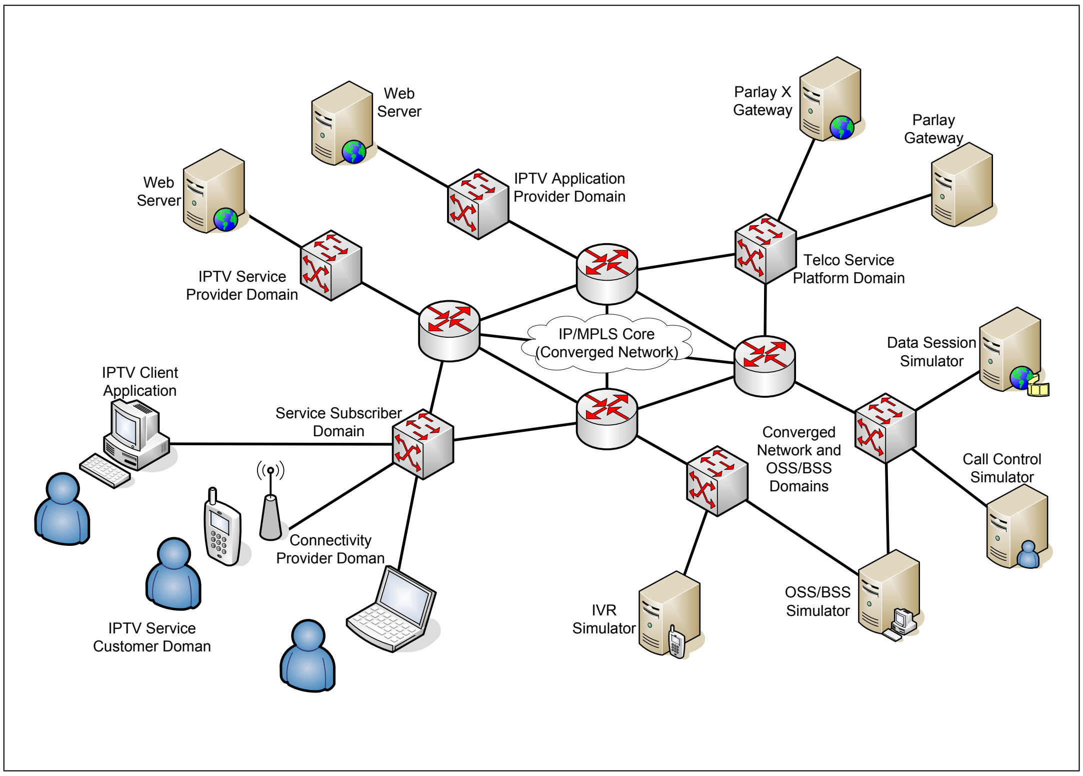


Figure 10.11: Deployment of SDP Implementation

10.6.2 Building, Deployment and Simulation

To implement applications and Parlay X web service we use the Java [47] programming language. Also, we use Java related products, such as the Netbeans [147] integrated development environment (IDE) to manage coding and Sun application server [148] to house the Parlay X web services, IPTV service provider application and IPTV provider application. The Java programming language provides an implementation of CORBA. Hence, the Parlay SCFs, SCS and network simulator (SCS clients) implementation is also Java-based.

The application, web service, SCF, SCS and SCS client software is implemented and deployed in the Wits University convergence laboratory [149]. The laboratory provides a diverse networking infrastructure to support the SDP and IPTV service software implementation. The software is distributed across the laboratories network equipment. This distribution is illustrated in the network diagram shown in *Figure 10.11*.

The diagram shows the division of the network, equipment and software across the SDP framework domains. We show the converged network supporting an IP-based transport network. Also, the network diagram shows various servers containing the SCS clients. These clients simulate OSS/BSS, call control, data session and IVR functionality. We use multiple web servers to host the IPTV service provider application, application provider

Client or Server	IP Address	Machine Name	Software
CORBA Naming Service	192.168.40.2	Zion - Windows Server	Java Runtime Environment (JRE)
Parlay Gateway (SCF/SCS)	192.168.40.2	Zion - Windows Server	JRE
Parlay X Gateway (Web Services)	192.168.40.3	Matrix - Windows Server	Java Application Server using JRE
Network Simulators	192.168.40.4	Neo - Windows Server	JRE
Service Provider Application	192.168.40.5	Trinity - Windows Server	Java Application Server using JRE
Application Provider Application	192.168.40.6	Agent Smith - Windows Server	Java Application Server using JRE
End-User Application	192.168.40.7	Morpheus - Windows Desktop	JRE

Table 10.1: SDP Deployment

application and Parlay X gateway. The Parlay gateway is contained within an application server. In addition, the diagram shows the end-users with fixed and mobile connectivity to a service subscriber.

We detail the equipment, their contained software and network addresses in *Table 10.1*. However, additional details on the SDP and IPTV service implementation and deployment is given in *Appendix C*. This appendix describes lessons learned using web services and CORBA for the SDP.

10.7 Results

The SDP implementation satisfies the business model and business relationship points. The SDP performs the service provider, service broker and content broker roles. The SDP takes on multiple roles since it uses its abundant services to easily satisfy these responsibilities and the associated business relationships. Also, by performing these roles the SDP remains the locus of *control* in the business model. For instance, the SDP manages external access to its services and therefore the converged network resources and capabilities. As a result, the SDP defines policies that manage business relationships with the external enterprises. Also, the SDP ensures adherence to these policies.

The SDP implementation uses technologies to realise the framework's layered and domain distributed GSOAs. These technologies include Parlay X web services, Parlay SCFs and IMS functional entities. The implemented GSOAs consistently map to reference points of the SDP reference model. The reference points promote *standardised* interactions between SDP, telco, end-users and external enterprised. However, to fully implement reference points some Parlay X and Parlay APIs were modified. In addition, some new interfaces were identified. Thus, the SDP implementation realises reference points by:

- using existing standard-based interfaces to implement a GSOA service interfaces;
- modifying existing standard-based interfaces to provide the required level of abstraction when implementing GSOA service interfaces; and
- creating new interfaces to implement GSOA service interfaces and promoting their standardisation.

The SDP implementation used all framework GSOA layers and distributed domains. However, the choice of technologies enables us to create alternative SDP architectures by collapsing layers or joining domains. For example, using Parlay X web services to overlap the intermediate and complex GSOA layers. The modification of the SDP framework into specific architectures illustrates its *versatility*. In addition, derived architectures retain the framework's concepts, principles and abstractions.

By implementing the SDP and IPTV application using Parlay X, Parlay and IMS, we have evaluated their use for standardising the SDP and providing various service interfaces. We illustrated many *gaps* within these technologies interfaces for the SDP. Hence, additional specification is required to ensure a complete standards-based SDP using Parlay X, Parlay and IMS. However, we may evaluate other technologies by mapping their standard-based interfaces to the framework's GSOAs service interfaces and defining an application that makes full use of these interfaces.

Therefore, the SDP framework provides a technology neutral foundation of concepts and abstractions that enables the creation of SDP architectures. These architectures are implementable using a variety of standard-based technologies that provide open service interfaces. In addition, the framework enables the *evaluation* of the technologies interfaces for a standards-based SDP implementation.

10.8 Summary

In this chapter we defined an SDP framework implementation that supports a IPTV service. We defined the requirements for the IPTV service using various use cases. The SDP implementation was based on a business model that defined various business entities, relationships and policies to support the IPTV service. A reference model was also used to identify reference points that the SDP implementation must formalise and implement. These reference points are elaborated as GSOAs contained in our SDP framework. We also used use cases to identify services that the GSOAs must provide to implement the reference points. The GSOAs were mapped to standard-based technologies. These technologies were the web services based Parlay X, CORBA-based Parlay and SIP-based IMS. These technologies provided implementations for the GSOAs middleware planes. Each technology provided open standard-based interfaces that mapped to GSOA service interfaces. However, we showed that these interfaces did not completely support the IPTV service and therefore did not aid in fulfilling business relationships and implementing reference points. As a result, we defined new web services and SCF interfaces. Also, we modified existing SCF interfaces to provide the required level of abstraction needed to support the IPTV service. We also implemented our own network simulator to manage Parlay and IMS interworking, since their interactions are not fully standardised. The SDP and IPTV service implementation was deployed on a physical network. The implementation proved the SDP framework concepts by showing the benefits of using the technology, implementation and distribution neutral framework to promote SDP standardisation.

Chapter 11

Contribution of our SDP Framework

This work set out describing the current trend of convergence between telco and IT-based enterprise networks. A major benefit of telecom-IT convergence is the development of a IT-based service platform that supports development, delivery and management of diverse services across telco and enterprise infrastructures. Current service platforms that support telecom-IT convergence are limited and based on proprietary technologies. In addition, the service platforms provide limited abstractions to hide complexities associated with convergence. We used the SDP concept to model a service platform framework that provides numerous abstractions and promotes SDP standardisation. The SDP framework was defined from different perspectives on standardised technologies, that are used in both telco and IT networks.

11.1 Summary of Results

From all perspectives we uncovered the concept of separation between service platforms and network functions. This separation ensured network intelligence is used independently of underlying network technologies. Also, the use of open standards for both service platforms and network functions maintain this separation.

We also uncovered numerous abstractions from all the perspectives. These include business models that justify external enterprise access to the SDP. By decomposing the business models we uncovered reference models with reference points. Reference points promote standardised interactions between the SDP and external enterprises, customers and the underlying network.

We found service abstractions to expose interfaces to implement reference points. Service interfaces expose diverse functionality to entities, while services implement logic to satisfy their interfaces. Service interfaces are technology, implementation and distribution neutral. We determined that these interface properties enabled diverse application implementations to access services.

Services and their interfaces are modeled into horizontal layers that are distributed across various domains. We found multiple layers are defined to structure services, based on their level of abstraction. Also, each layer exposes their service interfaces to higher layers. Hence, layers access each others functions by using their service interfaces. We also determined layers that expose their service interfaces to external enterprises represent points of integration between telco and IT-based infrastructures. This promotes standards-based telecom-IT convergence.

Another key abstraction found is the middleware plane that hides the distribution of layers and their services. The middleware plane abstracts numerous complexities associated with distribution, such as diverse software implementations, heterogenous computing platforms and unreliable communication links. We found that the middleware plane provides a wealth of services that are also standardised via their interfaces. These middleware services are used to support the operation and management of SDP services.

From the enterprise perspective we found a technology neutral container that embodies all the concepts and abstractions we uncovered. This container is the Generic Service Oriented Architecture (GSOA). From the converged perspective we used the GSOA to define a SDP architecture void of technology, implementation and distribution. The SDP architecture used GSOAs to implement reference points. Also, the GSOAs abstracted each other since they are horizontally layered and accessible via their service interfaces. The GSOA middleware planes abstracted the various distribution complexities.

We defined the SDP framework to represent a generic and extendable SDP architecture. The SDP framework is based on a business model and reference model, showing business relationships and reference points between SDP, telco, customers and external IT-using enterprises. To realise business relationships and reference points, the framework integrated multiple GSOAs. The framework used three GSOAs to model varying levels of services and their interfaces. In addition, we used GSOAs to model lower layer network functions and resources. Domains were used to distribute the framework GSOAs across customer, telco and enterprise areas. These domains related to the entities defined in the SDP business and reference models. However, the framework used the GSOA middleware planes to provide a bus-like abstraction, to hide distribution complexities.

We implemented the SDP framework to support an application that delivered content to end-users. The implementation was based on mapping technologies, with open standard-based service interfaces, to the GSOAs. In addition, we implemented the GSOA middleware planes using standard-based technologies, such as CORBA. To fully support the application and SDP we motivated the standardisation of new service interfaces. By mapping the technologies we showed how the SDP framework layers and domains may be altered to accommodate specific service interfaces. These modifications produced specific SDP architectures that inherited and maintained the framework concepts. This showed the versatility of the SDP framework.

11.2 Conclusion

This work contributes a way of thinking about the SDP, such that network and telecom-IT convergence complexities are simplified. This way of thinking is captured in the SDP framework. The framework represents generalised concepts extracted from many SDP perspectives, found in telecom, IT and Internet standards. The various SDP concepts are structured as abstractions. The SDP framework contains various layers of abstractions that are modeled as technology, implementation and distribution neutral building blocks called GSOAs.

The framework satisfied its objective to promote SDP standardisation. This is achieved by applying the framework to create technology, distribution and implementation independent SDP architectures. Also, the framework motivated the use of standard-based technologies with open interfaces to implement SDP architectures. By using standard-based technologies, the SDP provided open access to diverse IT-using enterprises and their applications. Standard-based technologies and interfaces enabled the SDP to consistently access converged network capabilities, including customer devices. Also, being standard-based ensured interoperability between various SDP implementations.

We proved the framework concepts by implementing a SDP using specific technologies with standard-based service interfaces. For the implementation we mapped existing service interfaces to most of the SDP reference points. However, additional standard-based service interfaces were required to implement the remaining reference points. The absence of standards for these SDP reference points illustrates the need and context for new standards to be developed. As a result, we created new interfaces to fully support the SDP and implement remaining reference points. Both existing and new service interfaces satisfied all reference points and therefore promoted SDP standardisation. Hence, the framework enabled us to evaluate standardised technologies and their service interfaces for use in a SDP implementation. In addition, the framework identified the need for new standard-based service interfaces to be developed and used in SDP implementations. The framework may also be used to test whether vendor solutions adequately implement the required SDP reference

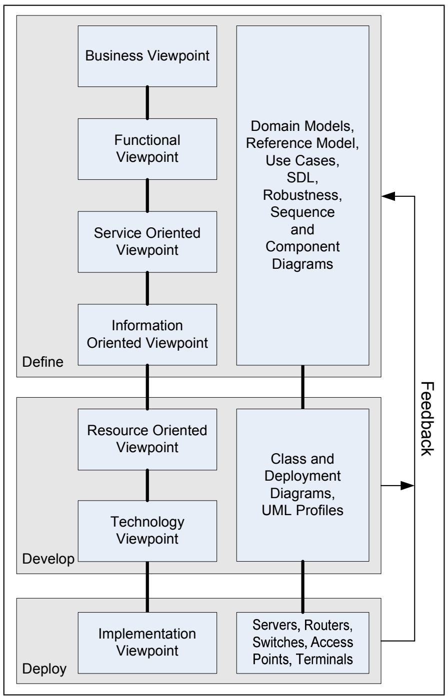


Figure 11.1: Example SDP Development Process using Viewpoints

points.

Therefore, the SDP framework provides a significant contribution towards full SDP standardisation.

11.3 Future Work

Based on the development of the SDP framework and proof of concept we recommend the following future work.

11.3.1 Development Process

We recommend the creation of a SDP development process that uses the generic SDP business model, reference model and framework to create standard-based SDP implementations. The development process should provide various viewpoints that enable a SDP architecture to created and implemented. We present an example illustration of a development process in *Figure 11.1*. The figure shows the viewpoints and associated modeling tools that may be used to capture viewpoint results.

In the figure, the business viewpoint determines the SDP's business model and business relationships. The functional viewpoint defines the functional and non-functional requirements of the SDP to support various external applications. A reference model may be used here to model and formalise SDP functions. The service-oriented viewpoint defines services and interfaces required to fulfil functional and business viewpoints. The viewpoint uses the SDP framework to structure a specific SDP architecture. The information viewpoint defines interface and policy details that satisfy reference points and business relationships. The resource-oriented viewpoint chooses specific standardised technologies to be mapped to the SDP architecture. The architecture may be manipulated to accommodate the technologies.

The technology viewpoint decomposes the complete SDP architecture into a more technology specific design. The technology specific design may include details on protocols, software environments, required equipment, networking requirements and management systems. The implementation viewpoint defines the physical deployment of the SDP implementation on a network.

11.3.2 Information Viewpoint

An information viewpoint on the SDP is essential since SDP services and external applications rely heavily on diverse forms of information. This information may be stored across the various SDP, telco, customer, enterprise and converged network.

The information viewpoint must identify various information abstractions that are scattered across the SDP framework. Each GSOA layer houses one or more information abstractions that are accessed by services and applications. As an example, we used CORBA middleware to house Parlay SCF location information that was accessed by web services across the intermediate GSOA.

The information viewpoint must also identify generic SDP services that provide access to the information abstractions. These information-oriented services must offer interfaces that simplify access to diverse information. Also, these SDP service interfaces must abstract access to lower GSOA information sources, such as an IMS HSS. As an example, we used the CORBA naming service to provide web services with access to Parlay SCF location information stored in the intermediate GSOA middleware plane.

11.3.3 Resource Oriented Viewpoint

The resource oriented viewpoint on the SDP enables the identification of existing standards-based technologies to implement SDP reference points. These technologies provide service interfaces that realise the abstract service interfaces identified in the service-oriented viewpoint. However, some reference points may not obtain suitable standards to implement their specifications. As a result, the resource oriented viewpoint enables us to identify standards that are required to fully implement a SDP. For instance, we recommend standards be defined to implement the R_{CS} reference point, since no suitable standards exist. These new standards would enable consistent communication between end-users and the SDP. Like the R_{CS} reference point, no suitable standard exists to implement R_{OS} reference point. These new standards will enable SDP and OSS/BSS communication via standard-based service interfaces.

The resource oriented viewpoint enables evaluation of standards that implement SDP reference points. For instance, the R_{TS} reference point may be implemented using the IMS and SIP. However, the IMS does not fully standardise communication between service platforms and network functional entities. As a result, a more complete IMS standard is required to provide the SDP with consistent and unified access to telco network resources and capabilities. Also, both Parlay and Parlay X implement the R_{DS} reference point using service interfaces that expose functionality to resource providers. These service interfaces are limited to content delivery and do not support content management. Additional service interfaces may be defined to strengthen the Parlay and Parlay X APIs. Therefore, gaps identified in existing standards used for the SDP represent additional research areas.

11.3.4 Creating Service Deliver Platforms

The SDP framework provides the foundation for any SDP architecture. These architectures may be defined by mapping standard-based technologies to the framework's GSOAs services, their interfaces, applications and supporting middleware planes. Also, these technology mappings may alter the framework's layers or domains to produce the final SDP architecture.

We motivate the design and implementation of various SDP architectures using the framework and mixtures of standard-based technologies. Also, we recommend evaluating implementations based on the appropriateness of their service interfaces within the SDP architecture. The SDP implementations may also result in the creation of specific applications that provide a new and innovative end-user services. These services may combine telco, IT and Internet functionality.

This proposed work may also contribute standard-based SDP architectures for the NGN.

11.3.5 Importance of Standardised Middleware

The middleware planes defined for the SDP framework's GSOAs provide necessary functions to abstract distribution complexities. The middleware planes extend functionality across the SDP framework domains. Hence, we motivate the standardisation of middleware technologies for the SDP. By promoting standardisation of middleware, their implementations become interoperable and even interchangeable.

The technologies used to implement the SDP framework middleware planes, must provide technology, implementation and distribution neutral services. These services hide and manage the various distribution complexities associated with a distributed system. In addition, these standard-based services contribute functionality to enhance SDP services and external enterprise applications.

Currently standards-based middleware technologies exist, such as CORBA. However, CORBA is complex to use and operate. In contrast the simple web services middleware, the ESB, is mostly proprietary. Hence, the ESB and its related technologies require additional standardisation. Also, interoperability between these different middleware technologies requires standardisation. However, middleware standardisation should be future proof, such that new middleware technologies easily interwork with legacy middleware.

References

[1] ITU-T, "*Specifications of Signalling System No. 7: Introduction to CCITT Signalling System No. 7*," Recommendation Q.700, March 1993.

[2] ITU-T, "*Principles of Intelligent Networks*," Recommendation I.312/Q.1201, October 1992.

[3] W. Hahn and R. Cowles, "*Government Involvement will be a Problem Through 2004*," Strategic Planning: Research Note SPA-21-4211, Gartner Group, 56 Top Gallant Road, Stamford, CT 06904, November 2003.

[4] A.C.Arnbak, "*Technology Trends and their Implications for Telecom Regulation*," in *Telecom Reform Principles, Policies and Regulatory Practices* (W.H. Melody, ed.), pp. 67–81, Technical University of Denmark, Lyngby, Denmark, 1997. ISBN: 87-7381-071-1.

[5] L. Corrigan, "*The Need for Comprehensive Partner Management in the Mobile Service Environment*," White Paper, Mobile Cohesion, 8b Weavers Court, Linfield Road, Belfast BT12 5GH.

[6] ITU-T, "*Definition of NGN*." Last accessed 01/12/2007, http://www.itu.int, November 2004.

[7] D. Booth *et al*, "*Web Services Architecture*," Working Group Note, W3C Web Services Architecture Working Group, February 2004.

[8] The Moriana Group, "*Service Delivery Platforms and Telecom Web Services*." A Moriana Thought Leader Report, Section A: Executive Summary, Last accessed 01/12/2007, http://www.morianagroup.com, June 2004.

[9] ITU-T, "*ITU-T Home Page*." Last accessed 01/12/2007, http://www.itu.int/ITU-T.

[10] B.Tilly and B.Briscoe, "*Metcalfe's Law is Wrong*," *IEEE Spectrum*, pp. 26–31, July 2006. Last accessed 01/12/2007, http://www.spectrum.ieee.org/jul06/4109.

[11] L. Rising and D. C. Schmidt, *Design Patterns in Communication Software*. The Pitt Building, Trumpington Street, Cambridge, UK: Press Syndicate of the University of Cambridge, 2001.

[12] A. Lundqvist, "*Connected Enterprise*," in *Business Summit*, Parlay Meeting, Osaka, Japan, November 2005.

[13] G. Parkins, "*A Service Orientated Architecture for Telecom Services*," in *KeyNote Session*, 9^{th} International Conference on Intelligence in Service Delivery Networks (ICIN) 2004, Bordeaux, France, October 2004.

[14] E. Christensen *et al*, "*Web Services Description Language (WSDL) 1.1*," Tech. Rep. 1.1, World Wide Web Consortium (W3C), March 2001.

[15] The OASIS UDDI Working Group, "*The OASIS UDDI Home Page*." Last accessed 01/12/2007, http://www.uddi.org.

[16] M. Gudgin *et al*, "*SOAP Version 1.2 Part 1: Messaging Framework*," Recommendation, W3C, June 2003.

[17] R. Fielding *et al*, "*Hypertext Transfer Protocol – HTTP/1.1*," Request For Comments (RFC) 2068, IETF Network Working Group, January 1997.

[18] B. Balabaskaran, "*Development of New Services with a Service Delivery Platform (SDP)*." Service Delivery Platforms and Telecom Web Services, A Moriana Group Thought Leader Report, Section B: Thought Leadership, Last accessed 01/12/2007, http://www.morianagroup.com, June 2004.

[19] The TINA Consortium (TINA-C), "*The TINA Home Page*." Last accessed 01/12/2007, http://www.tinac.com.

[20] The Parlay Group, "*The Parlay Home Page*." Last accessed 01/12/2007, http://www.parlay.org.

[21] ITU-T, "*Principles of Telecommunications Management Network*," Recommendation M.3010, May 1996.

[22] TeleManagement Forum, "*Telecommunications Operations Map*," Specification GB910, March 2000.

[23] Telemanagement Forum, "*eTOM The Business Process Framework*," Tech. Rep. GB921, October 2001.

[24] 3GPP, "*Technical Specification Group Services and Systems Aspects; Network Architecture (Release 7)*," Technical Specification TS 23.002 V7.0.0, December 2005.

[25] J. Soref and D. Troughton, “*The Real Meaning of Convergence*.” Service Delivery Platforms and Telecom Web Services, A Moriana Group Thought Leader Report, Section B: Thought Leadership, Last accessed 01/12/2007, http://www.morianagroup.com, June 2004.

[26] D. G. Messerschmitt, “*The Prospects for Computing Communications Convergence*,” in *Invited Papers*, Munchner Kreis Conference Vision 21: Perspectives for the Information and Communication Technology, Munich, Germany, November 1999.

[27] A. Henten *et al*, “*The Next Step for Telecom Regulation: ICT Convergence Regulation or Multisector Utilities Regulation*,” *The South African Journal of Information and Communication*, no. 3, 2003.

[28] F. Bosco *et al*, “*Service Broker: Identifying a New Role in the Mobile Services Provisioning Value Chain*,” in *Innovation in Business*, pp. 333–337, 9^{th} International Conference on Intelligence in Service Delivery Networks, October 2004.

[29] The DSL Forum, “*The DSL Forum Home Page*.” Last accessed 01/12/2007, http://www.dslforum.org.

[30] ETSI, “*Universal Mobile Telecommunications System; Requirements for the UMTS Terrestrial Radio Access system (UTRA) (UMTS 21.01 version 3.0.1)*,” Technical Report TR 101 111 V3.0.1, October 1997.

[31] N. Wilkinson, *Next Generation Network Services*. West Sussex, England: John Wiley and Sons, Ltd, 2002.

[32] The ATM Forum, “*The ATM Forum Home Page*.” Last accessed 01/12/2007, http://www.atmforum.com.

[33] The MPLS Forum, “*The MPLS Forum Home Page*.” Last accessed 01/12/2007, http://www.mplsforum.org.

[34] S. Deering *et al*, “*Internet Protocol Version 6 (IPv6) Specification*,” Request For Comment (RFC) 2460, IETF Network Working Group, December 1998.

[35] J. Postel, “*Transmission Control Protocol*,” Request For Comment (RFC) 793, Information Sciences Institute University of Southern California, September 1981.

[36] H. Schulzrinne *et al*, “*RTP: A Transport Protocol for Real-Time Applications*,” Request For Comments (RFC) 3550, IETF Network Working Group, July 2003.

[37] J. Rosenberg *et al*, “*SIP: Session Initiation Protocol*,” Request For Comments (RFC) 3261, IETF Network Working Group, June 2001.

[38] The International Packet Communication Consortium, "*The International Packet Communication Consortium Home Page*." Last accessed 01/12/2007, http://www.packetcomm.org.

[39] F. D. Ohrtman. JR, *Softswitch: Architecture for VoIP*. Networking Professional, New York. USA: McGraw-Hill, 2003.

[40] The SIGTRAN Working Group, "*The SIGTRAN Home Page*." Last accessed 01/12/2007, http://www.sigtran.org.

[41] N. Greene *et al*, "*Media Gateway Control Protocol Architecture and Requirements*," Request For Comments (RFC) 2850, IETF Network Working Group, April 2000.

[42] P. Calhoun *et al*, "*Diameter Base Protocol*," Request For Comments (RFC) 3588, IETF Network Working Group, September 2003.

[43] ETSI, "*Digital Cellular Telecommunications System (Phase 2+); Customised Applications for Mobile network Enhanced Logic (CAMEL) Phase 2; Stage 2*," Technical Specification 101 441 v.6.7.0, ETSI, August 2000.

[44] C. Abarca *et al*, "*Service Architecture*," TINA-C Deliverable 5.0, Telecommunications Information Networking Architecture (TINA) Consortium, June 1997.

[45] C. Abarca *et al*, "*Network Resource Architecture*," TINA-C Deliverable 3.0, Telecommunications Information Networking Architecture (TINA) Consortium, February 1997.

[46] M. Corporation, "*.Net Framework Home Page*." Last accessed 01/12/2007, http://msdn2.microsoft.com/en-us/netframework/.

[47] Sun Microsystems, "*Java Home Page*." Last accessed 01/12/2007, http://java.sun.com/.

[48] TMF, "*NGOSS Lifecycle and Methodology*," Tech. Rep. GB927, Release 4.3, Version 1.3, TeleManagement Forum, November 2004.

[49] Sun Microsystems, "*JSLEE and the JAIN Initiative*." Last accessed 01/12/2007, http://java.sun.com/products/jain/, 2007.

[50] Open Mobile Alliance (OMA), "*OMA Home Page*." Last accessed 01/12/2007, http://www.openmobilealliance.org, 2007.

[51] OMA, "*OMA Service Environment*," Specification V1.0.2, August 2005.

[52] *Appium Home Page*, "*Appium*." Last accessed 01/12/2007, http://www.appium.com.

[53] Erricsson, "*Service Delivery Platforms*." Last accessed 01/12/2007, http://www.ericsson.com/, 2006.

[54] IBM, "*IBM Service Provider Delivery Environment: A Technical Overview*," IBM Telecommunication Industry White Paper, http://ibm.com/industries/telecom/spde, May 2005.

[55] Microsoft, "*Enabling Service Delivery using the Microsoft Connected Framework*," White Paper, http://www.microsoft.com, January 2005.

[56] SDP Alliance, "*Service Delivery Platform*." Last accessed 01/12/2007, http://sdpalliance.mobilitydatasystems.com/.

[57] Hewlett-Packard, "*Service Delivery Platform*." Last accessed 01/12/2007, http://www.hp.com, 2007.

[58] CERN Engineering Data Management Service, "*CERN Engineering Data Management Service Glossary*." Last accessed 01/12/2007, http://cedar.web.cern.ch/CEDAR/glossary.html, 2001.

[59] Oxford University Press, "*Oxford English Dictionary*." Last accessed 01/12/2007, http://www.oup.com.

[60] A. P. Sage and J. E. Armstrong Jr, *Introduction to Systems Engineering*. Wiley Series In Systems Engineering, New York, USA: John Wiley and Sons, Inc, 2000.

[61] S. Graupner *et al*, "*A Framework for Organizing Complex Systems*," Internal Paper HPL-2001-24, Hewlett-Packard Company, HP Laboratories, Palo Alto, February 2001.

[62] B. S. Blanchard and W. J. Fabrycky, *Systems Engineering and Analysis*. Industrial and Systems Engineering, New Jersey, USA: Prentice Hall, third edition ed., 1998.

[63] P. Checkland, *Systems Thinking, Systems Practice*. New York, USA: John Wiley and Sons, Ltd, 1981.

[64] B. Liskov and J. Guttag, *Abstraction Specification in Program Development*. The MIT Electrical Engineering and Computer Science Series, The MIT Press, Cambridge, Massachusetts: The MIT Press with the McGraw-Hill Book Company, second ed., 1987.

[65] D. G. Messerschmitt, "*Complexity Management: A Major Issue for Telecommunications*," (Stranford University, Palo Alto, CA), pp. 169–180, International Conference on Communications, Computing, Control and Signal Processing, June 1995.

[66] H. E. Hanrahan, "*A Comparative Study of Telecommunication Architectures: Methodology and Case Studies*," South African Telecommunication Network and Applications Conference (SATNAC) 2003, September 2003.

[67] I. R. Hoos, *Systems Analysis in Public Policy*. University of California Press, 1972.

[68] L. Bass *et al*, *Software Architecture in Practice*. Addison-Wesley, second ed., 2003.

[69] J. Bosch, *Design and Use of Software Architectures*. Software Engineering / Software Architecture, Addison-Wesley, 2000.

[70] G. Mustapic *et al*, "*Real World Influences on Software Architecture- Interviews with Industrial Systems Experts*," IEEE Working Conference on Software Architectures, Oslo, Norway, June 2004.

[71] K. Smolander, "*Four Metaphors of Architecture in Software Organizations: Finding out the Meaning of Software Architecture*," in *ISESE '02: Proceedings of the 2002 International Symposium on Empirical Software Engineering*, pp. 211–221, IEEE Computer Society, October 2002.

[72] ISO/IEC 10746-1, "*Open Distributed Processing Reference Model Part 1: Overview*," ITU-T Recommendation, May 1995.

[73] J. Miller *et al*, "*MDA Guide Version 1.0.1*," Specification omg/2003-06-01, The Object Management Group, June 2003.

[74] K. Smolander *et al*, "*What Is Included in Software Architecture? A Case Study in Three Software Organizations*," in *ECBS '02: Proceeding of the 9th IEEE International Conference on Engineering of Computer-Based Systems*, pp. 131–138, IEEE Computer Society, 2002.

[75] ITU-T, "*General Aspects of the Intelligent Network Application Protocol*," Recommendation Q.1208, September 1997.

[76] J. Thörner, *Intelligent Networks*. Artech House Telecommunications Library, 685 Canton Street, Norwood, MA, 02062: Artech House Publishers INC., 1994.

[77] J. Zuidweg, *Next Generation Intelligent Networks*. Telecommunications, Artech House INC,685 Canton Street, Norwood, MA 02062: Artech House Publishers, 2002.

[78] T. Magendanz and R. P. Zeletin, *Intelligent Netwoks*. Brekshire House, 168-173 High Holborn, London WCIV 7AA, UK: International Thomson Computer Press, 1996.

[79] R. T. Sanders, "*Service-Centred Approach to Telecom Service Development*," in *Adaptable Networks and Teleservices*, IFIP WG6.7 Workshop and EUNICE Summer School, Norwegian University of Science and Technology, Trondheim, Norway, September 2002.

[80] M. Chapman and S. Montesi, "*Overall Concepts and Principles of TINA*," TINA-C Deliverable 1.0, Telecommunications Information Networking Architecture (TINA) Consortium, February 1995.

[81] G. Coulouris, J. Dollimore and T. Kindberg, *Distributed Systems: Concepts and Design*. International Computer Science Series, Addison-Wesley, third ed., 2001.

[82] The Object Management Group (OMG), "*The OMG Home Page*." Last accessed 01/12/2007, http://www.omg.org.

[83] O.M.G, "*Common Object Request Broker Architecture: Core Specification*," Formal Version 3.0.3, March 2004.

[84] M. Mampaey, "*TINA for Services and Advanced Signalling and Control in Next Generation Networks*," *IEEE Communications Magazine*, pp. 104–110, October 2000.

[85] C. Smith, "*Applying TINA-C Service Architecture to the Internet and Intranets*," in *Global Convergence of Telecommunications and Distributed Object Computing*, pp. 4–12, TINA 97, Santiago, Chile, November 1997.

[86] Z. Lozinski, "*Parlay: A Report to Members*." Last accessed 01/12/2007, http://www.parlay.org, November 2003.

[87] R. Jain *et al*, *Programming Converged Networks*, vol. Wiley-Interscience of *Computer Science*. John Wiley and Sons, Inc, 2005.

[88] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 5)*," ETSI Standard 202 915-1 V1.1.1, April 2004.

[89] M. Unmehopa, *Parlay/OSA from Standards to Reality*. John Wiley and Sons, Ltd, 2006.

[90] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework (Parlay 5)*," ETSI Standard 203 915-3 V1.2.1, January 2007.

[91] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 2: Generic Call Control SCF (Parlay 4)*," ETSI Standard 202 915-4-2 V1.2.2, August 2003.

[92] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction SCF (Parlay 4)*," ETSI Standard 202 915-5 V1.2.1, August 2003.

[93] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility SCF (Parlay 4)*," ETSI Standard 202 925-6 V1.2.1, August 2003.

[94] S. Guo, F. Yang and J. Chen, "*Open Integrated Intelligent Network: A universal Service Platform*," in *Circuits and Systems*, pp. 74–77, Asia-Pacific Conference on Circuits and Systems, Tianjin, China, 2000.

[95] K. Chung and Y. Choi, "*An Interworking Mechanism between SCFs and Protocols in the Open Service Gateway*," in *Communications*, pp. 1–5, Asia-Pacific Conference on Communications, Busan, Korea, August 2006.

[96] The Parlay Group: Web Services Working Group, "*Parlay Web Services Architecture Comparison*," Tech. Rep. 1.0, October 2002.

[97] J. Popoff, "*Parlay/OSA 101*," in *Education Seminar*, Parlay/OSA Americas Conference, Boston USA, October 2005.

[98] The Parlay Group: Web Services Working Group, "*Parlay Web Services Overview*," Tech. Rep. 1.0, October 2002.

[99] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 2: Third Party Call (Parlay X 2)*," Standard 202 391-2 v1.2.1, December 2006.

[100] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 11: Audio Call (Parlay X 2)*," Standard 202 391-11 v1.2.1, December 2006.

[101] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 9: Terminal Location (Parlay X 2)*," Standard 202 391-9 v1.2.1, December 2006.

[102] A. Pras, B. Beijnum and R. Sprenkels, "*Introduction to TMN*," CTIT Technical Report 99-09, University of Twente, the Netherlands, 1999.

[103] *ITU-T*, "*Management Framework For Open Systems Interconnection (OSI) For CCITT Applications*," Recommendation x.700, September 1992.

[104] M. Azmoodeh *et al*, "*Evaluation of TMN Status, OSS/TMN Deployment*," Main Report 812-GI, EURESCOM, December 1999.

[105] TMF, "*TeleManagement Forum SID Web Page*." Last accessed 01/12/2007, http://www.tmforum.org/browse.aspx?catID=1684.

[106] TMF, "*TeleManagement Forum TNA Web Page*." Last accessed 01/12/2007, http://www.tmforum.org/browse.aspx?catID=1685.

[107] ITU-T, "*Information Technology, Open Systems Interconnection, Common Management Information Protocol Specification for CCITT Applications*," Recommendation x.711, ITU-T, October 1991.

[108] H. Hanrahan, *Network Convergence*. John Wiley and Sons, Ltd, 2007.

[109] TMF, "*NGOSS and Shared Information Model*," Last accessed 01/12/2007, http://www.tmforum.org/browse.aspx?catID=1684, Data Sheet, TeleManagement Forum, November 2005.

[110] R. Swinarski, "*TeleManagement Forum: NGOSS Architecture Overview*," Presentation, TMF, April 2003. http://www.tmforum.org.

[111] J. L. Fleck, "*Overview of the Structure of the NGOSS Architecture*," White Paper, Hewlett Packard, May 2003.

[112] J. Strassner *et al*, "*TMF White Paper on NGOSS and MDA*," White Paper 1.0, TeleManagement Forum, Novemeber 2003.

[113] D. Sprott, "*Service Oriented Architecture: An Introduction for Managers*," CBDI Forum and Roadmap Report, Last accessed 01/12/2007, http://www.cbdiforum.com, May 2005.

[114] E. A. Marks, "*Build a Better Enterprise Application*," *Network Magazine*, vol. 19 (8), August 2004.

[115] K. Channabasavaiah and K. Holley, "*Migrating to a Service-Oriented Architecture*," White Paper G224-7298-00, IBM, April 2004.

[116] B. Silver, "*Agile to the Bone*," *Intelligent Enterprise*, vol. 8, February 2005.

[117] T. Andrews *et al*, "*Business Process Execution Language for Web Services*," Specification 1.1, BEA Systems, May 2003.

[118] S. Craggs, "*Best-of-Breed ESBs*," White Paper, Enterprise Application Integration (EAI) Industry Organization, June 2003.

[119] W. Andrews, "*SOA has Impact on Application Development Outsourcing*," Strategic Planning: Research Note (SPA-22-5494), Gartner Group, Last accessed 01/12/2007, http://www.gartner.com, April 2004.

[120] J. Egli, "*IMS: A Journey not a Leap*." http://www.convergedigest.com, Last accessed 01/12/2007, September 2005.

[121] 3rd Generation Partnership Project (3GPP), "*3GPP Home Page*." Last accessed 01/12/2007, http://www.3gpp.org.

[122] 3GPP, "*Technical Specification Group Services and System Aspects; Service requirements for the Internet Protocol (IP) multimedia core network subsystem; Stage 1 (Release 7)*," Technical Specification TS 22.228 V7.3.0, December 2005.

[123] 3GPP, "*Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 7)*," Technical Specification TS 23.228 V7.2.0, December 2005.

[124] C. Gourraud *et al*, "*The IMS Application Layer(s)*," in *Challenges for the IMS*, pp. 2–7, 10^{th} International Conference on Intelligence in Service Delivery Networks (ICIN), Bordeaux, France, May 2006.

[125] 3GPP, "*Technical Specification Group Core Network; IP Multimedia (IM) Session Handling; IM Call Model; Stage 2 (Release 6)*," Technical Specification TS 23.218 V6.2.0, September 2004.

[126] 3GPP, "*Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API) Mapping for Open Service Access; Part 4: Call Control Service Mapping; Subpart 4: Multiparty Call Control ISC (Release 6)*," Technical Specification TS 29.998-04-4 V6.0.4, December 2004.

[127] 3GPP, "*Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API) Mapping for OSA; Part 4: Call Control Service Mapping; Subpart 1: API to CAP Mapping (Release 6)*," Technical Specification TS 29.998-04-1 V6.0.0, December 2004.

[128] 3GPP, "*Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API) Mapping for OSA; Part 5: User Interaction Service Mapping; Subpart 1: API to CAP Mapping (Release 6)*," Technical Specification TS 29.998-05-1 V6.0.0, December 2004.

[129] 3GPP, "*Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API) Mapping for OSA; Part 6: User Location - User Status Service Mapping to MAP (Release 6)*," Technical Specification TS 29.998-06 V6.0.0, December 2004.

[130] V. Radhakrishnan *et al*, "*PIAF: An Application Framework for Unlocking IMS Engendered Network Capabilities*," in *Programming Models for the IMS*, pp. 165–170, 10^{th} International Conference on Intelligence in Service Delivery Networks (ICIN), Bordeaux, France, May 2006.

[131] M. Brenner, "*From Collision to Cooperation*," *IEC News Letter*, vol. 2, August 2007, Last accessed 01/12/2007, http://www.iec.org.

[132] G. Deckers, "*Cost Down, Revenues Up: SDP Business Case*," in *Business Aspects of Service Convergence*, pp. 178–183, 10^{th} International Conference on Intelligence in Service Delivery Networks (ICIN), Bordeaux, France, May-June 2006.

[133] TMF OSS/J Home Page, "*OSS through Java Initiative*." Last accessed 01/12/2007, http://www.tmforum.org, 2006.

[134] O. M. G. (OMG), "*Unified Modelling Language (UML) Resource Page*." Last accessed 01/12/2007, http://www.uml.org.

[135] The Parlay Group: The Parlay X Working Group, "*Parlay APIs 4.0 Parlay X Web Services White Paper*," Tech. Rep. 1.0, December 2002.

[136] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 6: Payment (Parlay X 2)*," ETSI Standard 202 391-6 V1.2.1, December 2006.

[137] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 12: Charging SCF (Parlay 5)*," ETSI Standard 203 915-12 V1.2.1, Januray 2007.

[138] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 7: Account Management (Parlay X 2)*," ETSI Standard 202 391-7 V1.2.1, December 2006.

[139] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 11: Account Management SCF (Parlay 5)*," ETSI Standard 203 915-11 V1.2.1, Januray 2007.

[140] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 4: Short Messaging (Parlay X 2)*," ETSI Standard 202 391-4 V1.2.1, December 2006.

[141] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction SCF (Parlay 5)*," ETSI Standard 203 915-5 V1.2.1, Januray 2007.

[142] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 3: Multi-Party Call Control SCF (Parlay 5)*," ETSI Standard 203 915-4-3 V1.2.1, Januray 2007.

[143] ETSI, "*Open Service Access (OSA); Parlay X Web Services; Part 14: Presence (Parlay X 2)*," ETSI Standard 202 391-14 V1.2.1, December 2006.

[144] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 14: Presence and Availability Management SCF (Parlay 5)*," ETSI Standard 203 915-14 V1.2.1, Januray 2007.

[145] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 8: Data Session Control SCF (Parlay 5)*," ETSI Standard 203 915-8 V1.2.1, Januray 2007.

[146] ETSI, "*Open Service Access (OSA); Application Programming Interface (API); Part 13: Policy Management SCF (Parlay 5)*," ETSI Standard 203 915-13 V1.2.1, January 2007.

[147] Netbeans, "*Netbeans Home Page*." Last accessed 01/12/2007, http://www.netbeans.org.

[148] Sun Microsystems, "*Sun Java System Application Server 9.1*." Last accessed 01/12/2007, http://www.sun.com/software/products/appsrvr/.

[149] Wits University Convergence Lab, "*Wits Convergence Lab Home Page*." Still under construction, http://www.ee.wits.ac.za/comms.

Appendix A

Mapping to the SDP Framework

To show the flexibility of the SDP framework, we map both standard-based and proprietary technologies onto its GSOAs. In these mappings we show gaps in existing technologies that must be standardised to fully support the SDP. These gaps may motivate additional research into the standardisation of the SDP.

A.1 Standards-based Architectures

In *Table A.1*, we map IN/TMN architectures [2, 21] onto the SDP framework. In *Table A.2*, we map the TINA architecture [44, 45] onto the SDP framework. In *Table A.3*, we map the eTOM framework [23] onto the SDP framework. In *Table A.4*, we map the Java API for Integrated Networks (JAIN) [49] architecture onto the SDP framework. The JAIN architectures has not been discussed in this work. However, it provides an architecture that combines both IN and Parlay.

The standard-based technologies provide fuller mappings than the proprietary solutions. The IN/TMN mapping shows a legacy SDP focusing on complex services and the required functions. TINA provides an end-to-end mapping onto the SDP framework that extensively uses middleware to support a variety of rich components. The eTOM mapping looks limited, but focuses on providing simple services to abstract the entire OSS/BSS domain and its intersecting layers. The JAIN mapping provides a SDP, by implementing existing telecom and IT standards using the Java programming language and tools. This mapping also shows the integration of Java specific components and technologies with the SDP.

A.2 Product-based Architectures

In *Figure A.1*, we show the Ericsson SDP architecture [53]. In *Table A.5*, we map the Ericsson SDP architecture onto the SDP framework. In *Figure A.2*, we show the HP SDP architecture [57]. In *Table A.6*, we map the HP SDP architecture onto the SDP framework. In *Figure A.3*, we show the IBM SDP architecture [54]. In *Table A.7*, we map the IBM SDP architecture onto the SDP framework. In *Figure A.4*, we show the Microsoft SDP architecture [55]. In *Table A.8*, we map the Microsoft SDP architecture onto the SDP framework.

All the proprietary technology mappings show limited details, since they use mixtures of standardised and proprietary technologies to implement framework layers. Also, many of these proprietary solutions provide implementations for the service platform, OSS/BSS domains and their intersecting layers. Hence, these solutions do not provide a complete end-to-end mapping.

	Generic Service Delivery Platform Architecture						
$\downarrow$ *Layers/Domains* $\rightarrow$	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps					Services and features		
Simple Services							
Simple Service Middleware							
Intermediate Apps							
Intermediate Services							
Intermediate Service Middleware							
Complex Apps					Global service logic		
Complex Services					SIBS		
Complex Service Middleware							
Service Functions				IWF	SCF, SDF, SRF, SSF	SMF (OSFs)	SMAF, SCEF
Service Function Middleware							
Resource Functions			CCF, CCAF	CCF	CCAF, CCF	NEF (OSFs)	
Resource Function Middleware							
Resources	Telephone		Switch	Switch	SCP, SDP, SSP, SRP	Network Elements	
Resource Middleware					INAP	SNMP	

Table A.1: Mapping IN/TMN onto the SDP Framework

Generic Service Delivery Platform Architecture							
↓ $Layers/Domains$ →	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps							
Simple Services							
Simple Service Middleware							
Intermediate Apps	asUAP, ssUAP				USM		USM
Intermediate Services	PA	IA, UA			PUSM, SSM, SF, UA		PUSM, SSM, SF, UA
Intermediate Service Middleware	CORBA-based Distributed Processing Environment (DPE)						
Complex Apps							
Complex Services	TCSM	CSM, CSMF			CSM, CSMF		CSM, CSMF
Complex Service Middleware	CORBA-based Distributed Processing Environment (DPE)						
Service Functions			CCF, CC, FCC				
Service Function Middleware	CORBA-based Distributed Processing Environment (DPE)						
Resource Functions	TLA, CP		LNC, TM, TCM, CP				
Resource Function Middleware	CORBA-based Distributed Processing Environment (DPE)						
Resources	Heterogenous and Distributed Networking Equipment						
Resource Middleware	Various Protocols						

Table A.2: Mapping TINA onto the SDP Framework

	Generic Service Delivery Platform Architecture						
↓ *Layers/Domains* →	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps						Business Processes	
Simple Services						Web Services	
Simple Service Middleware						ESB	
Intermediate Apps						J2EE Services	
Intermediate Services						CORBA, RMI, ...	
Intermediate Service Middleware							
Complex Apps							
Complex Services						Resource Adaptors	
Complex Service Middleware							
Service Functions							
Service Function Middleware							
Resource Functions							
Resource Function Middleware							
Resources						CRM, ERP, ...	
Resource Middleware							

Table A.3: Mapping eTOM onto the SDP Framework

Generic Service Delivery Platform Architecture							
$\downarrow$ $Layers/Domains$ $\rightarrow$	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps	Web Service App						Web Service App
Simple Services					Web Services		
Simple Service Middleware	SOA-based middleware, such as an ESB						
Intermediate Apps	Java Apps				EJBs		
Intermediate Services	J2ME, J2SE				J2EE		
Intermediate Service Middleware	Various middleware may be used such as ESB, CORBA, RMI, ...						
Complex Apps					Service Building Blocks (SBB)		
Complex Services	MIDP, CLDC				Numerous Resource Adaptors		
Complex Service Middleware	JVM				JSLEE		
Service Functions							
Service Function Middleware							
Resource Functions							
Resource Function Middleware							
Resources	Terminal				Network Elements	CRM, ERP, ...	
Resource Middleware	Various Protocols						

Table A.4: Mapping JAIN onto the SDP Framework

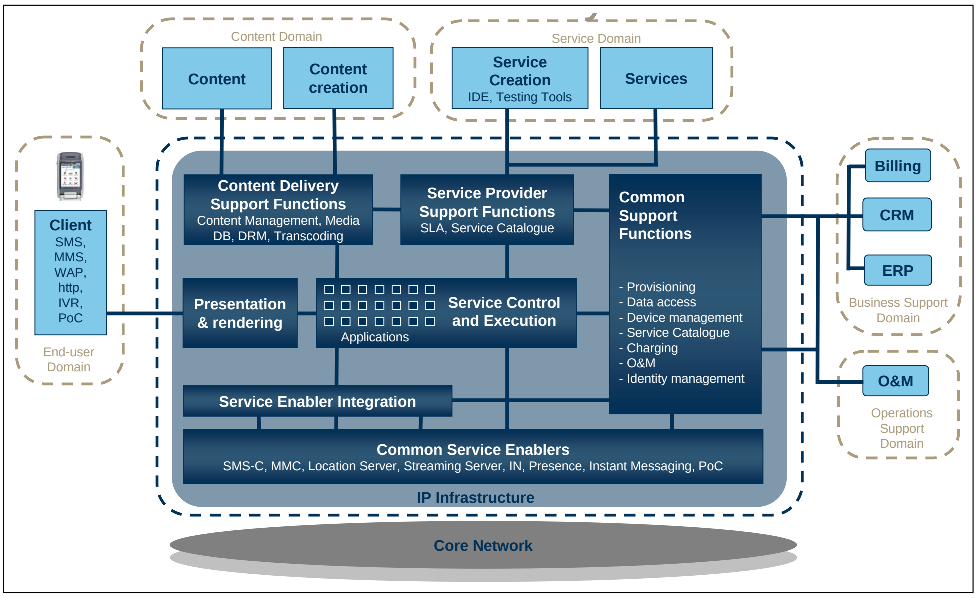


Figure A.1: Ericsson SDP Architecture

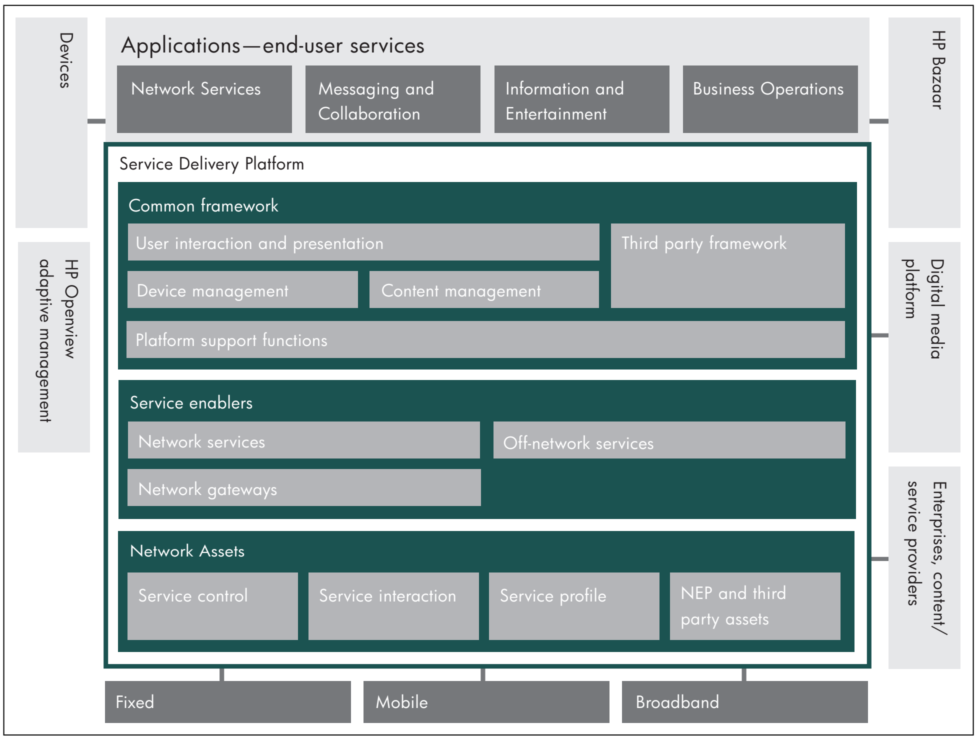


Figure A.2: Hewlett Packard SDP Architecture

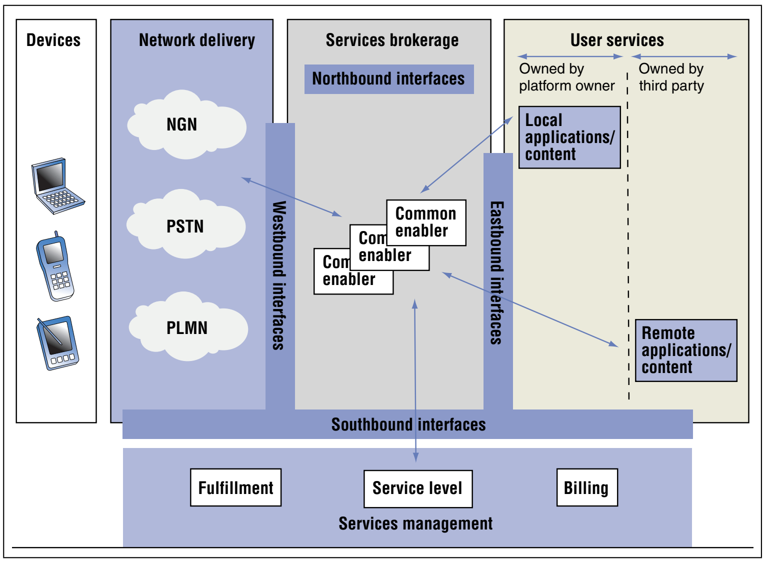


Figure A.3: IBM SDP Architecture

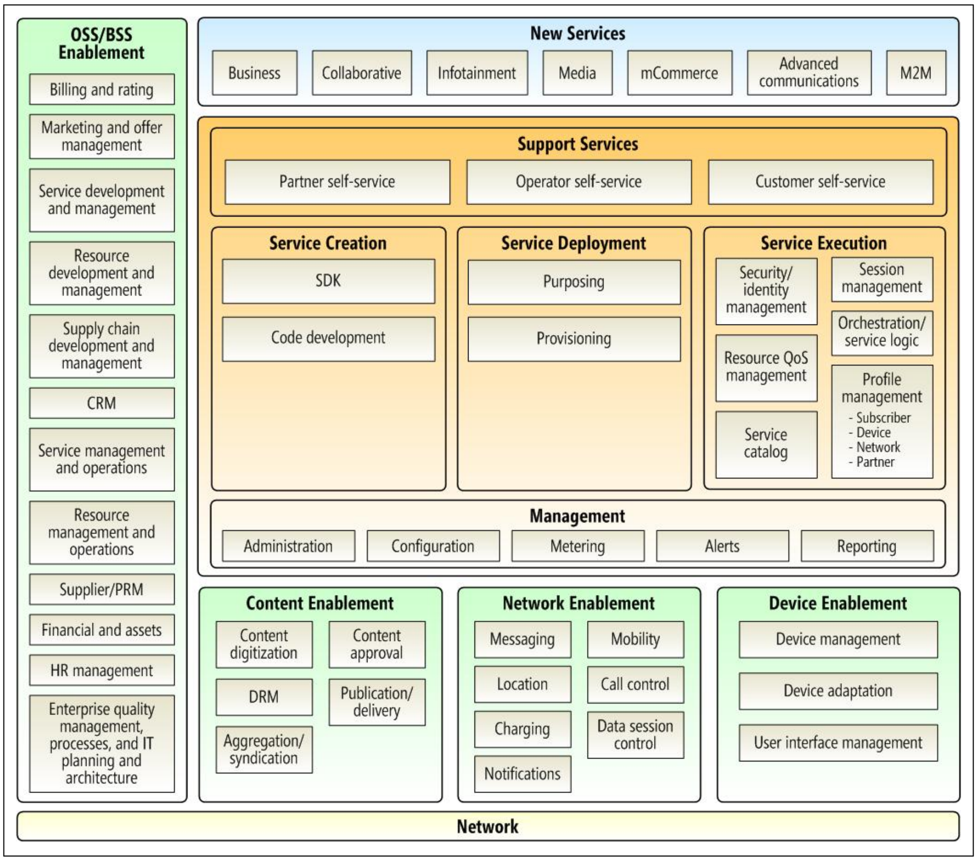


Figure A.4: Microsoft SDP Architecture

	Generic Service Delivery Platform Architecture						
↓ *Layers/Domains* →	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise App Services
Simple Apps					Content Delivery/Service Provide Support Functions		
Simple Services							
Simple Service Middleware							
Intermediate Apps	Client Apps						
Intermediate Services	Client Services				Presentation, Rendering, Service Control &	Common Support Functions	
Intermediate Service Middleware					Execution		
Complex Apps							
Complex Services					Service Enabler Integration		
Complex Service Middleware							
Service Functions					Common Service Enablers		
Service Function Middleware							
Resource Functions							
Resource Function Middleware							
Resources						Billing, CRM, ERP, O&M	
Resource Middleware							

Table A.5: Mapping the Ericsson SDP Architecture onto the SDP Framework

Generic Service Delivery Platform Architecture							
↓ $Layers/Domains$ →	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps	Services						Applications
Simple Services					Common Framework		
Simple Service Middleware	Mixtures of standards and proprietary technologies						
Intermediate Apps							
Intermediate Services					Service Enablers		
Intermediate Service Middleware	Mixtures of standards and proprietary technologies						
Complex Apps							
Complex Services					Service Enablers		
Complex Service Middleware	Mixtures of standards and proprietary technologies						
Service Functions							
Service Function Middleware					HP OpenCall, Convergent Charging, Service Management,...		
Resource Functions							
Resource Function Middleware							
Resources							
Resource Middleware							

Table A.6: Mapping the Hewlett Packard SDP Architecture onto the SDP Framework

Generic Service Delivery Platform Architecture							
$\downarrow$ $Layers/Domains$ $\rightarrow$	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps	Applications				Local Apps/Content		Remote Apps/Content
Simple Services					Common Enablers with interfaces		
Simple Service Middleware					Mixtures of standards and proprietary technologies		
Intermediate Apps							
Intermediate Services					Common Enablers with interfaces		
Intermediate Service Middleware					Mixtures of standards and proprietary technologies		
Complex Apps							
Complex Services					Common Enablers with interfaces		
Complex Service Middleware					Mixtures of standards and proprietary technologies		
Service Functions							
Service Function Middleware							
Resource Functions							
Resource Function Middleware							
Resources	Devices						
Resource Middleware							

Table A.7: Mapping the IBM SDP Architecture onto the SDP Framework

Generic Service Delivery Platform Architecture							
↓ $Layers/Domains$ →	End User	Subscriber	Connectivity Provider	Converged Network	Service Platform	OSS/BSS	Enterprise
Simple Apps					New Services		New Services
Simple Services					Support Services	Enablement	
Simple Service Middleware					SOA-based ESB		
Intermediate Apps							
Intermediate Services					Service Creation, Deployment, Execution	Enablement	
Intermediate Service Middleware					Undefined but provides Management capabilities		
Complex Apps							
Complex Services					Content, Network, Device Enablement	Enablement	
Complex Service Middleware							
Service Functions							
Service Function Middleware							
Resource Functions							
Resource Function Middleware							
Resources							
Resource Middleware							

Table A.8: Mapping the Microsoft SDP Architecture onto the SDP Framework

Appendix B

Message Sequence Charts for IPTV Service

In this appendix we show additional message sequence diagrams for the SDP proof of concept. We show interactions between the Parlay SCS and IMS functional entities. As a result, we map SCS API invocations to SIP messages. We use our extended SCS API since its current interface is not fully standardised. We also show the Parlay SCFs invoking the extended SCS API.

We illustrate interactions between Parlay X web services and the Parlay SCFs. These interactions occur across web service and SCF APIs. We extend specific SCFs to provide additional capabilities to satisfy the SDP proof of concept. These capabilities are reflected in new or enhanced SCF APIs. As a result, we abstract access to these SCF APIs by defining new web service APIs. The new web service APIs ensure an appropriate level of abstraction is exposed to external IT-using enterprises.

The following message sequences only specify the application, web service, SCF and SCS API methods. We do not provide detail on the method parameters, since these may be rich data structures. Also, for the SCS and IMS mappings we only provide recommendations for SIP message extensions. We do not give detail on the contents of the new SIP messages.

B.1 Web Service and SCF Interactions

In this section we provide details on the interactions between external providers, end-users and the SDP.

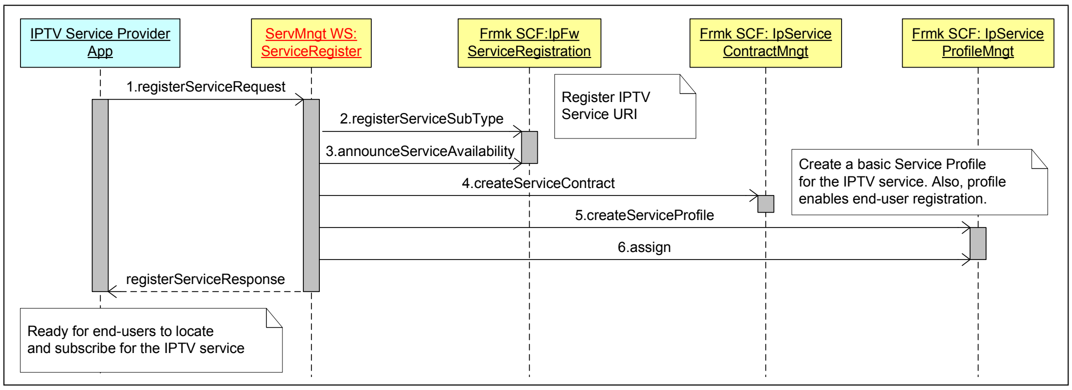


Figure B.1: Registration of IPTV Service

B.1.1 IPTV Service Registration

Currently, the Parlay framework provides limited subscription-oriented APIs. These APIs enable one or more SCFs to be registered with the framework. However, we enhance these APIs to enable service providers to register their end-user services with the framework. In addition, we abstract the richness of these APIs into a newly defined Parlay X service management web service API. The web service API simplifies access to the framework service management capabilities. In *Figure B.1*, we show the following interactions involving the extended framework APIs and new service management web service APIs:

1. The service provider wishes to register its IPTV service with the SDP. The service provider may pass the URI of its IPTV service as one of the method parameters. In addition, the service provider may provide information or a word list describing the service its offering.

2. The web service requests the framework to register this service. The framework detects an end-user service is being registered and not a SCF.

3. The web service requests the framework to make this service visible to other entities, such as service brokers or end-users.

4. The web service requests the framework to create a service contract for the service provider.

5. The web service requests the framework to create a basic service profile for the IPTV service, within its service contract.

6. The web service requests the framework to associates the services provider's contract and profile. The web service returns successfully to the service provider.

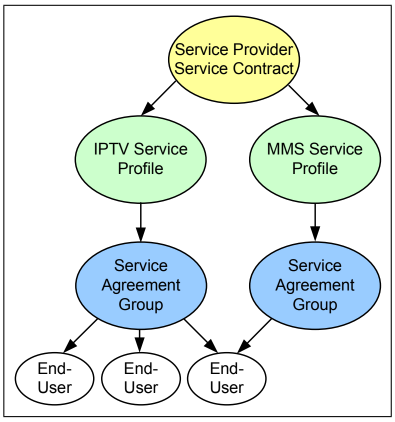


Figure B.2: Service Provider and End-User Registration Model

In *Figure B.1*, we reuse the framework's service contract and service profile abstractions. However, we reuse these abstractions differently, since we are registering end-user services and not SCFs. We show the modified service contract and service profile abstractions in *Figure B.2*. In the figure, the service contract contains service provider registration information. This contract recognises the service provider as a provider of end-user specific services. The service provider's IPTV service is also registered within the service contract as an IPTV service profile. The service provider may have multiple service profiles for different end-user services. For example, a service provider may have a single service contract that contains an IPTV service profile, MMS service profile and telephony service profile.

The IPTV service profile contains a service agreement group (SAG). The SAG is used to house one or more end-user registrations. These end-users have successfully registered to use the IPTV service. The end-user registration contains information such as username, password, account details and phone number.

B.1.2 End-User Service Location, Registration and Deregistration

In *Figure B.3*, we use the service management web service to abstract the service discovery capabilities of the framework SCF. The frameworks' IpServiceDiscovery API is used used to locate other SCFs. We extend this API to provide end-user service discovery. This enables end-users to locate the registered IPTV service.

In the figure, we extend the framework's client application registration capabilities to include end-user registrations. These capabilities are exposed by a new IpEndUserManagement API. This interface inherits some methods from the framework's existing IpClientAppManagement API. The new API enables the service provider to populate its IPTV service profile with a SAG and end-user subscriptions. We also extend the framework to

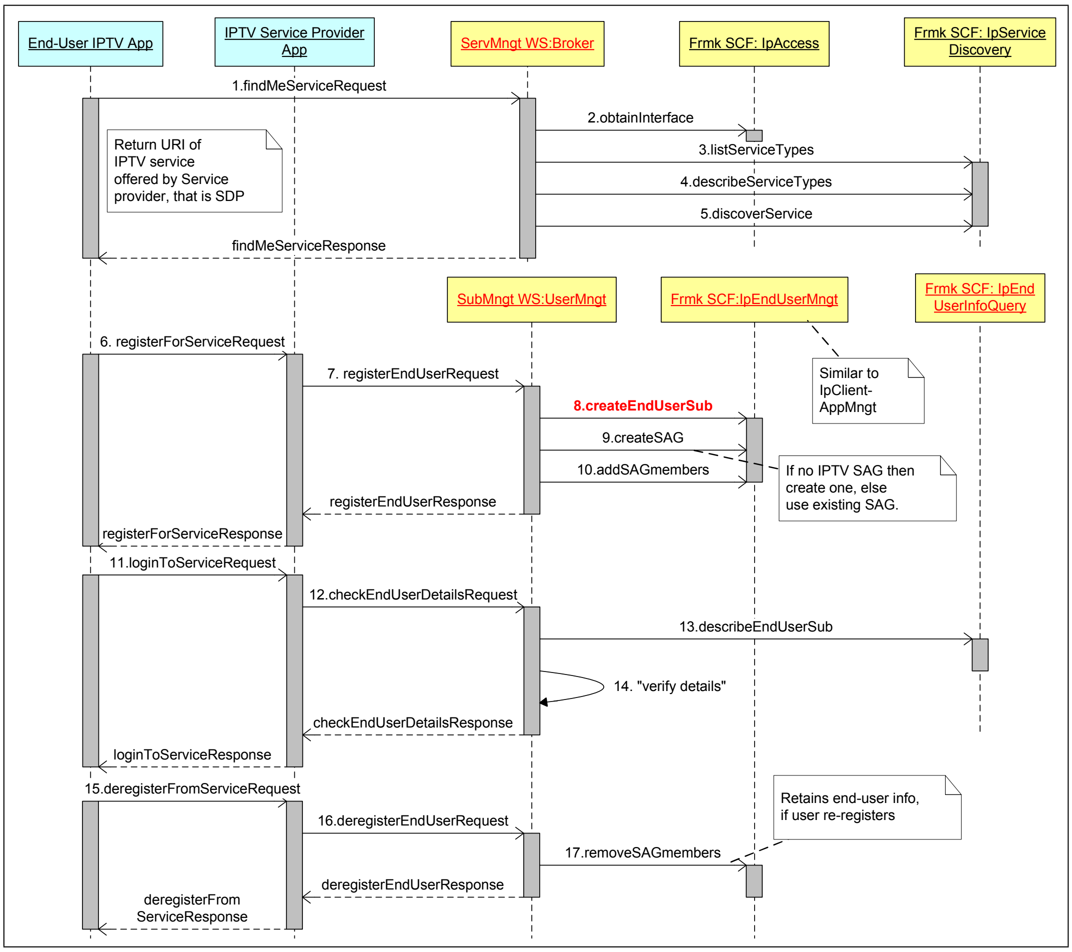


Figure B.3: End-User Message Sequences

provide an IpEndUserInfoQuery API that enables end-user subscription information to be retrieved. We abstract access to the new framework APIs by defining a new Parlay X based subscription management web service API.

In *Figure B.3* the interactions are as follows:

1. The end-user requests the service management web service to locate a television service. The end-user may provide additional details on the service it requires.

2. The web service requests access to the service discovery interface on the framework.

3. The web service requests the framework to list services that match the end-user's request. The framework determines an end-user service list is required and not a SCF list.

4. The web service requests the framework to provide additional details on a specific service that it has found to match the user request. This service being the IPTV service.

5. The web service requests the framework to provide the location of the IPTV service. This may represent a URI. The web service returns the location to the end-user.
6. Using the URI, the end-user registers with the IPTV service provider by providing personal information.
7. The service requests the subscription management web service to store the registration information in its service contract and IPTV service profile.
8. The web service requests the framework to create a new end-user subscription.
9. If this is the first subscription, the web service requests the framework to create a new SAG to hold the end-user subscription. The SAG is stored as part of the IPTV service profile.
10. The web service requests the framework to add a new end-user subscription to the SAG. Once completed the web service returns successfully to the service provider, who forwards the result to the end-user.
11. The end-user logs into the IPTV service by providing his/her username and password.
12. The service requests the subscription management web service to verify the end-user's username and password stored in his registration. This information is contained in the IPTV service profile SAG.
13. The web service requests the framework to obtain the end-user subscription information from the IPTV service profile SAG.
14. The web service extracts the end-user's username and password and verifies if it is correct. The web service returns the result to the service provider, who forwards the result to the end-user.
15. At some point the end-user may deregister with the IPTV service provider.
16. The service requests the subscription management web service to deregister the end-user by removing him/her from the SAG.
17. The web service requests the framework to remove the end-user subscription from the IPTV service profile SAG. The web service returns a successful deregistration result to the service provider, who forwards the result to the end-user.

B.1.3 Media Provider Registration

Similar to the previous service registration, service location and end-user subscription, we reuse the Parlay framework's capabilities to mange content registration, content location

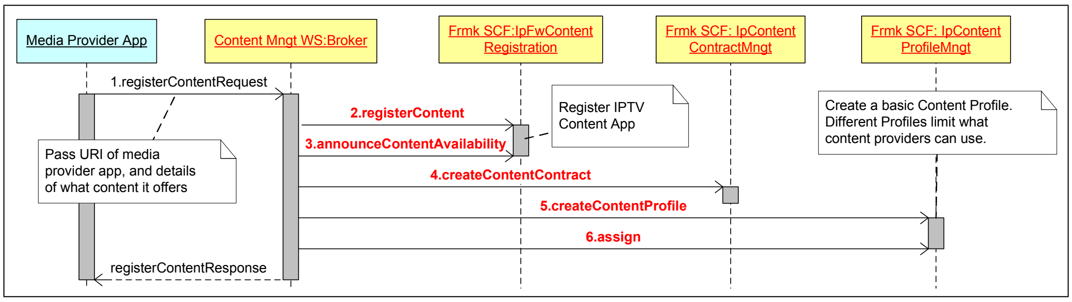


Figure B.4: Registration of Media Provider

and application provider registration.

In *Figure B.4*, we extend the frameworks SCF registration capabilities to include content registration. These new capabilities are exposed by new IpFwContentRegistration, IpContentContractManagement and IpContentProfileManagement APIs. These new APIs are similar to the framework's existing IpFwServiceRegistration, IpServiceContractManagement and IpServiceProfileManagement APIs. The new APIs provide methods that enable media providers to register themselves as content providers with the framework. Also, media providers may register their various content sources with the framework. We abstract access to the framework's new content registration capabilities using a new Parlay X based content management web service API.

In *Figure B.4* the interactions are as follows:

1. The media provider wishes to register its diverse television content with the SDP. The service provider passes a URI of an application that manages access to its content sources. In addition, the service provider may provide information or a word list describing its offered content.

2. The web service requests the framework to register this media provider as a content provider, using its application and content list. The framework detects a content provider is being registered and not a SCF.

3. The web service requests the framework to make this content visible to other entities, such as application providers.

4. The web service requests the framework to create a content contract for the media provider.

5. The web service requests the framework to create a basic content profile, within its content contract, describing the media provider's offered television content.

6. The web service requests the framework to associates the media provider's content

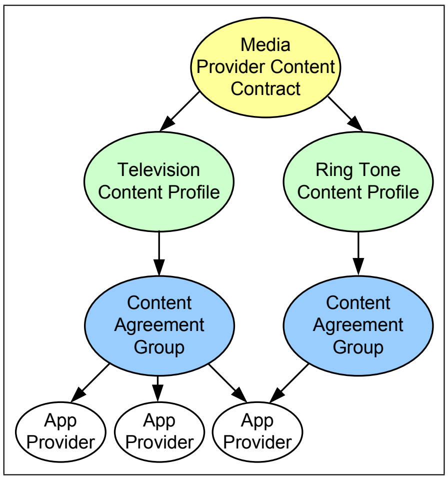


Figure B.5: Media Provider and Application Provider Registration Model

contract and television content profile. The web service returns a successful outcome to the media provider.

In *Figure B.4*, we again reuse the framework's service contract and service profile abstractions. However, these abstractions are now content-oriented, since we are registering content and not SCFs. We show the content contract and content profile abstractions in *Figure B.5*. In the figure, the content contract contains media provider registration information. This implies the media provider is registered with the framework as a provider of television content. The media provider's television content is also registered within the content contract as a content profile. The profile contains the location of the application providing access to the television content source(s). The media provider may have multiple content profiles for different content types. For example, a media provider may have a single content contract that contains a television content profile, picture content profile and ring tone content profile.

The television content profile contains a content agreement group (CAG). The CAG is used to house one or more application provider registrations. These application providers have successfully registered to use the television content. The application provider registration contains information such as username, password, account details and address.

B.1.4 Application Provider Content Location, Registration and Deregistration

In *Figure B.6*, we enhance the framework's SCF discovery capabilities to include content discovery. The content has already been registered using its new content registration APIs,

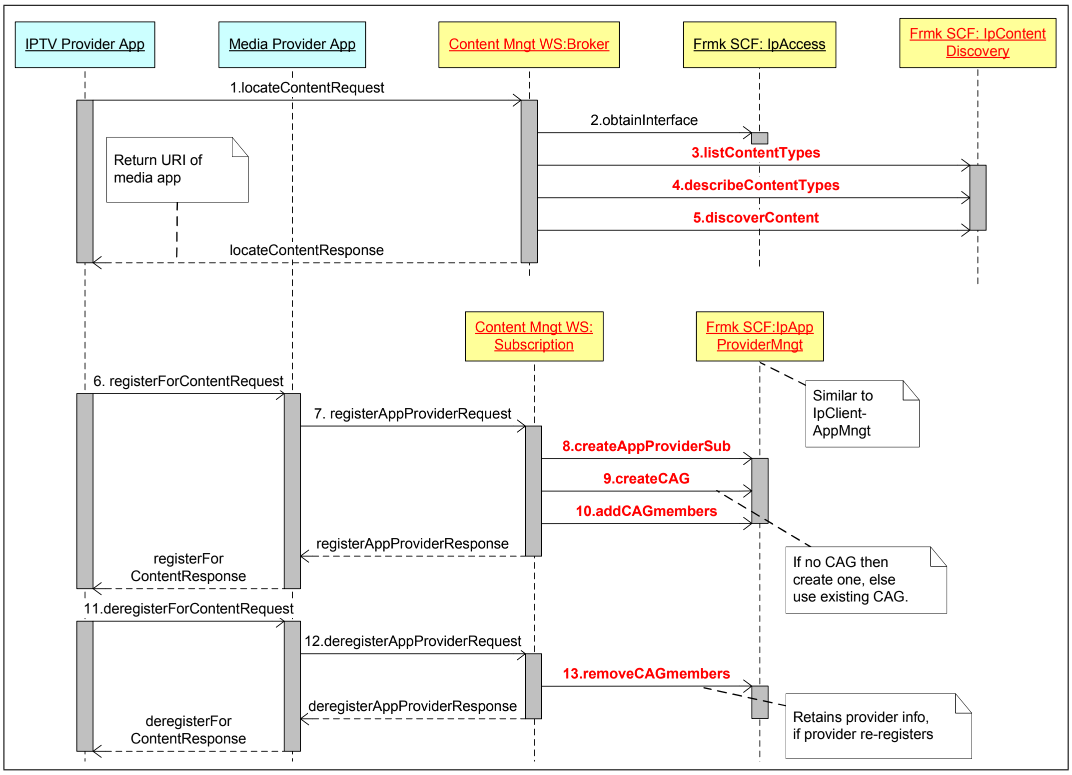


Figure B.6: Application Provider Message Sequences

discussed previously. The content discovery capabilities are exposed by a new IpContent-Discovery API. This API is similar to the framework's existing IpServiceDiscovery API.

In the figure, we also extend the framework's client application registration capabilities to include application provider registrations. These capabilities are exposed by a new IpApp-ProviderManagement API. This interface is similar to the framework's existing IpClientAppManagement API. The new API enables the media provider to populate its television content profile with a CAG and application provider subscriptions. We abstract access to the new framework APIs by using the content management web service API.

The *Figure B.6* we show the following interactions:

1. The IPTV application provider requests the content management web service to locate television content offered by a media provider. In its request, the application provider may provide specific details on the content it requires.

2. The web service requests access to the new content discovery interface on the framework.

3. The web service requests the framework to list content that match application provider's request.

4. The web service requests the framework to provide additional details on specific content that it has found to match the application provider's request. This content being the television content.
5. The web service requests the framework to return the location of the media provider application, that will provide access to the television content. This location is represented as an URI. The web service returns the location to the application provider.
6. Using the URI, the application provider registers with the media provider. The application provider provides various information in this request.
7. The media provider application requests the content management web service to store the registration information in its television content profile CAG.
8. The web service requests the framework to create a new application provider subscription.
9. If this is the first subscription, the web service requests the framework to create a new television content profile CAG to hold the subscription information.
10. The web service requests the framework to add the new application provider subscription to the television content profile CAG. Once completed the web service returns the result to the media provider, who forwards the result to the application provider.
11. At some point the application provider may deregister with the media provider.
12. The media provider requests the content management web service to deregister the application provider by removing its subscription from the television content profile CAG.
13. The web service requests the framework to remove the application provider subscription from the media provider's television content profile CAG. The web service returns a successful deregistration result to the media provider, who forwards the result to the application provider.

B.1.5 SDP and Enterprise Policy Management

In *Figure B.7*, we define a new Parlay X based policy management web service API. This API abstracts access to the existing policy management SCF API. Application providers, media providers and service providers use the web service to view SDP policies. Also, the web service API enables external entities to create their own policies to manage access to their services and content. The figure shows the following interactions:

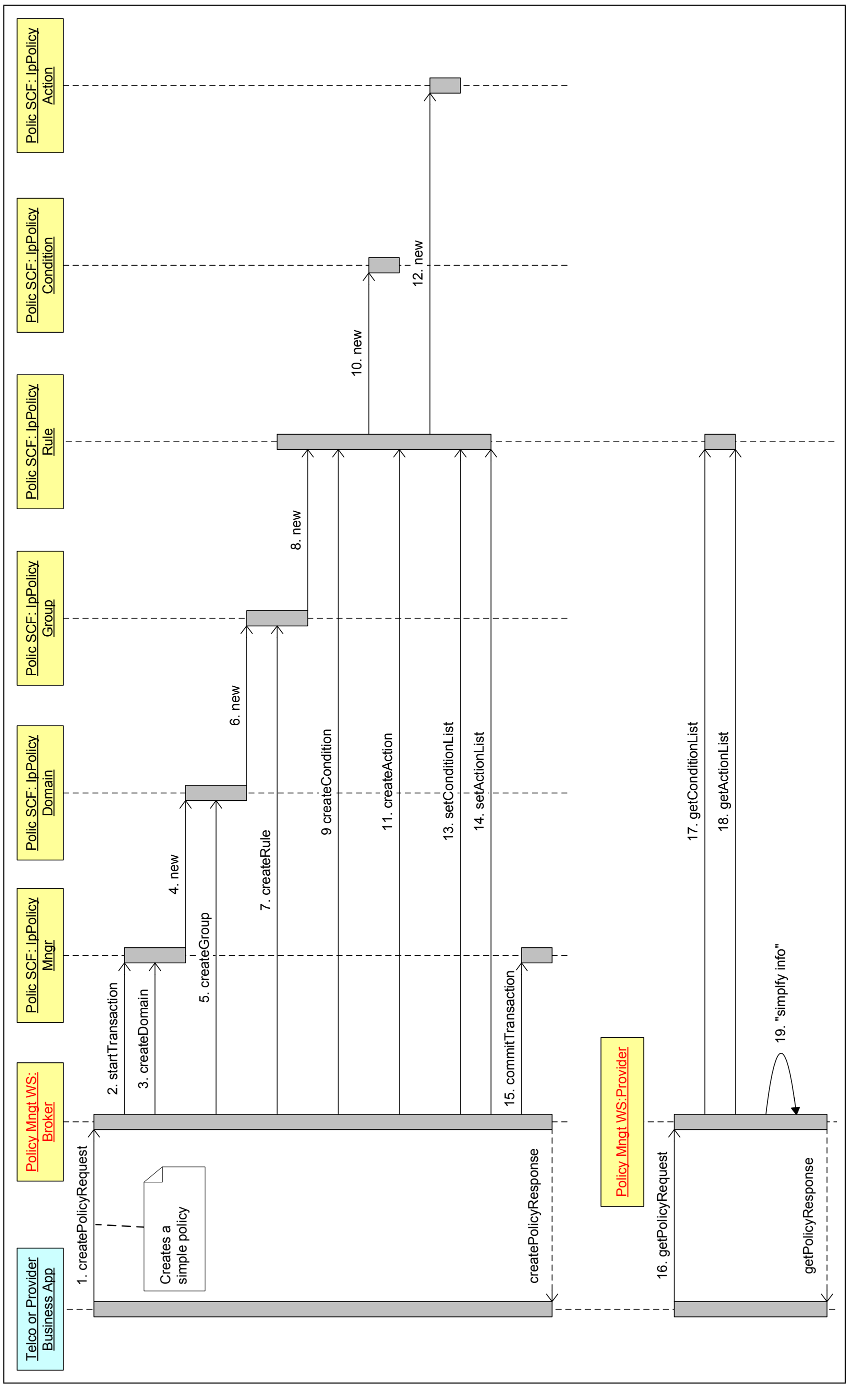


Figure B.7: Create and Obtain Policy Information

1. An enterprise or SDP application requires a new policy to be created, via the policy management web service. The request includes structured information describing the new policy.

2. The web service starts a new policy transaction with the policy SCF.

3. The web service creates a new policy domain, relating to the entity that requires the policy.

4. A new policy domain object is created.
5. The web service creates a new policy group, within the domain, to hold the details of the new policy.
6. A new group object is created.
7. The web service creates a new rule, within the group, to define the policy.
8. A new rule object is created.
9. The web service creates a new condition, within the rule.
10. A new condition object is created.
11. The web service creates a new action that is invoked when the condition is satisfied.
12. A new action object is created.
13. A new condition list is created that defines the conditions to be satisfied to invoke a specific action.
14. A new action list is created that defines the actions to invokes when conditions are satisfied.
15. The web service commits this policy to the SCF. A response is returned to the external entity or telco.
16. An external or SDP application requests information about a policy that has been created. This request is made on a different API exposed by the policy management web service.
17. The web service invokes the SCF to obtain condition details that constitute the requested policy.
18. The web service invokes the SCF to obtain corresponding actions details that relate to the policy conditions.
19. The web service constructs a simplified representation of the policy details and returns this information to the requesting entity.

In *Figure B.8*, we use the policy management web service API to remove policies. The web service API enables policies to only be deleted by those who defined them. These policies are managed by the policy management SCF. The figure shows the following interactions:

1. The requesting entity requires the deletion of a specific policy.

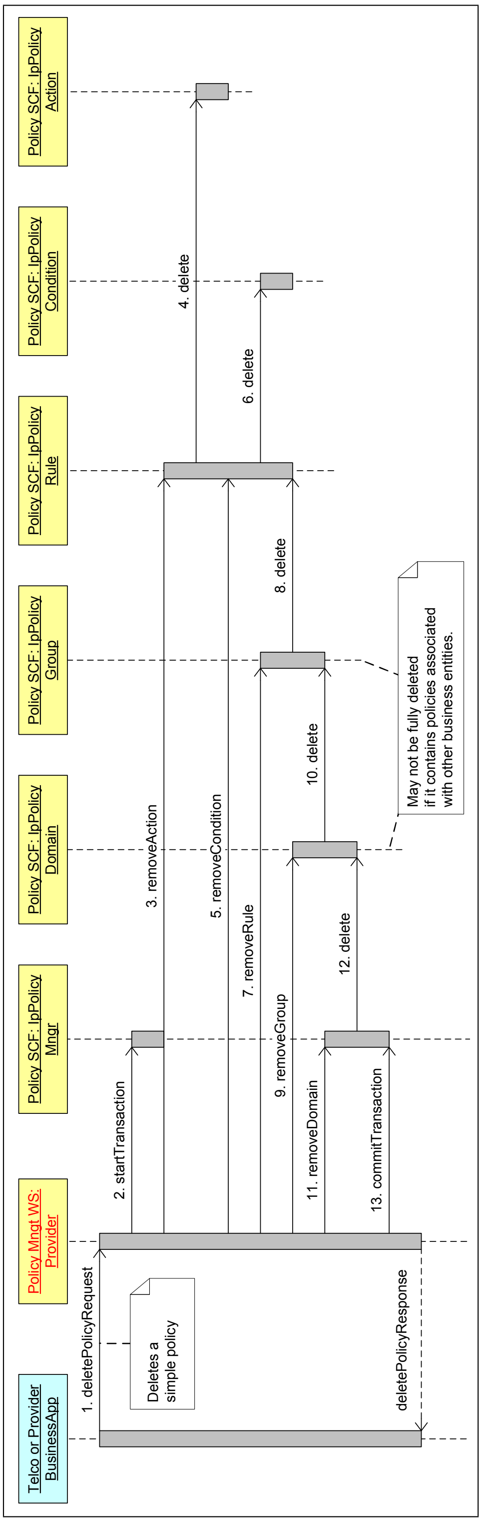


Figure B.8: Remove a Policy

2. The web service starts a new policy transaction with the policy SCF.

3. The web service removes the actions associated with the policy conditions.

4. The actions are removed and the action object is destroyed.

5. The web service removes the conditions associated with the policy.
6. The conditions are removed and the condition object is destroyed.
7. The web service removes the rule associated with the policy.
8. The rule object is destroyed.
9. The web service removes the policy group associated with policy to be deleted.
10. The policy group object is destroyed, if their are no other rules store in that group.
11. The web service removes the domain associated with the policy group.
12. The domain object is destroyed, if there are no more groups stored in that domain.
13. The web service commits these removal actions to the SCF. A response is returned to the external entity or telco.

B.1.6 End-User Account Management

In *Figure B.9*, we use the existing Parlay X based account management web service API. This API abstracts access to the account management SCF API. The IPTV service provider uses this web service to manage their end-users' accounts. The figure shows the following interactions:

1. The end-user requires his/her account balance.
2. The IPTV service provider requests the end-user's balance from the account management web service.
3. The web service creates a new callback object. The web service requests this object to obtain the end-user's balance.
4. The web service callback object requests the end-user's balance from the account management SCF.
5. The SCF requests the SCS to obtain the end-user's balance.
6. The SCS returns the end-user's balance to the SCF.
7. The SCF returns the end-user's balance to the web service callback object.
8. The callback object returns the end-user's balance to the web service. The web service returns the result to the service provider, who forwards it to the end-user.

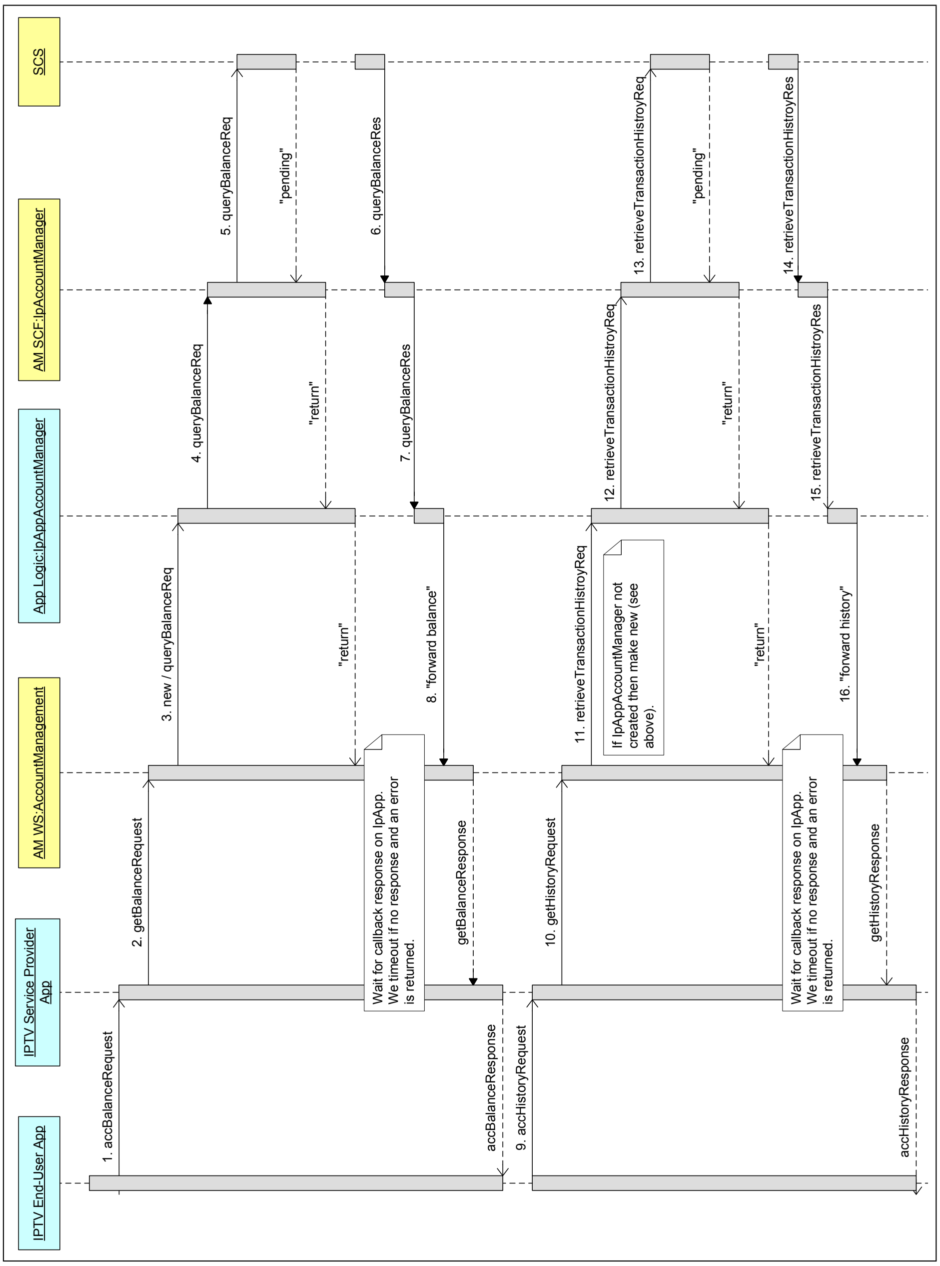


Figure B.9: Obtain Account Balance and History

9. The end-user requires his/her account history.
10. The IPTV service provider requests the end-user's account history from the account management web service.
11. The web service reuses its existing callback object or creates one if none is available. The web service requests its callback object to obtain the end-user's account history.
12. The web service callback object requests the end-user's account history from the account management SCF.
13. The SCF requests the SCS to obtain the end-user's account history.

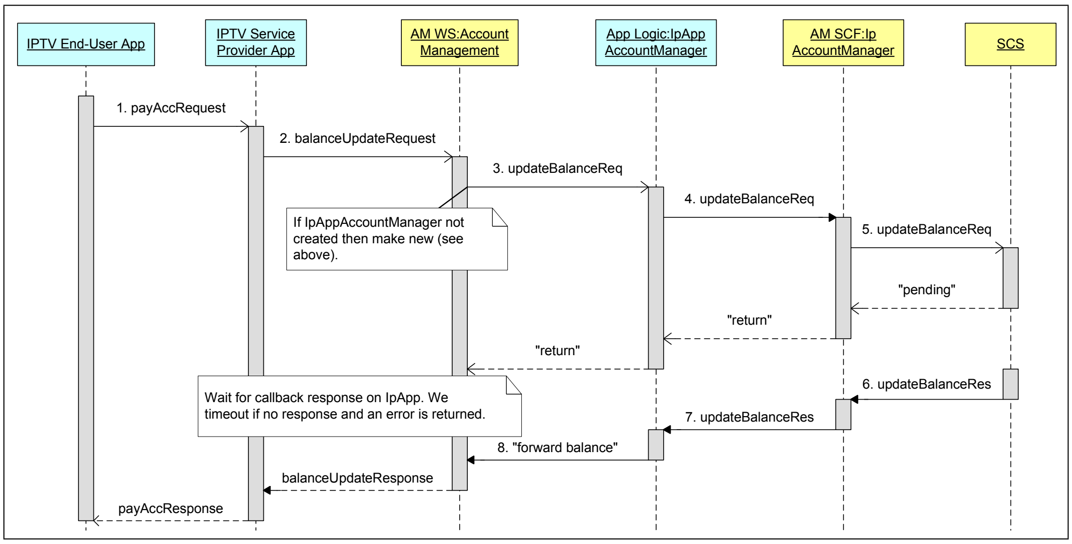


Figure B.10: Pay End-User Account

14. The SCS returns the end-user's account history to the SCF.

15. The SCF returns the end-user's account history to the web service callback object.

16. The callback object returns the end-user's account history to the web service. The web service returns the result to the service provider, who forwards it to the end-user.

B.1.7 End-Use Payments

In *Figure B.10*, we again use the account management web service API. The IPTV service provider uses this web service to manage their end-user payments. The figure shows the following interactions:

1. The end-user requires to make a payment towards his account using his/her credit card or IPTV payment vouchers.

2. The IPTV service provider requests the web service to update its end-users account balance.

3. The web service creates or uses an existing callback object to update the end-user's account balance.

4. The web service callback object requests the account management SCF to update the end-user's account balance.

5. The SCF requests the SCS to update the end-user's account balance.

6. The SCS returns a successful update has been made to the end-user's account balance.
7. the SCF returns the successful result to the web service callback object.
8. the callback object returns the result to the web service. The web service returns the result to the service provider, who forwards it to the end-user.

B.1.8 End-User Pay Per View Part 1

In *Figure B.11*, we use the existing Parlay X payment web service API. This web service API simplifies access to the Payment SCF. The IPTV service provider uses the payment web service to charge end-user accounts for viewing special IPTV content. The figure shows the following interactions:

1. The end-user requests a the television guide from the service provider. The guide will indicate content that requires payment and those that are free to view.
2. The service provider requests a list of available television content from the IPTV service provider.
3. The application provider requests the content list from the media provider. The media provide returns the list to the application provider. The list is returned to the service provider, who generates the television guide and returns the guide to the end-user.
4. The end-user selects a television programme to watch. This programme is not free and requires payment.
5. The service provider requests the payment web service to charge the end-user's account for the content.
6. The web service creates a callback object to manage asynchronous responses from the payment SCF.
7. The web service requests the payment SCF to create a charging session to modify the end-user's account.
8. The new charging session object is created in the SCF.
9. We add this method to the SCF API, to allow the web service access to the newly created charging session's interface.
10. The web service informs the charging session to reserve an amount from the end-user's account, such that he/she can watch the selected content.

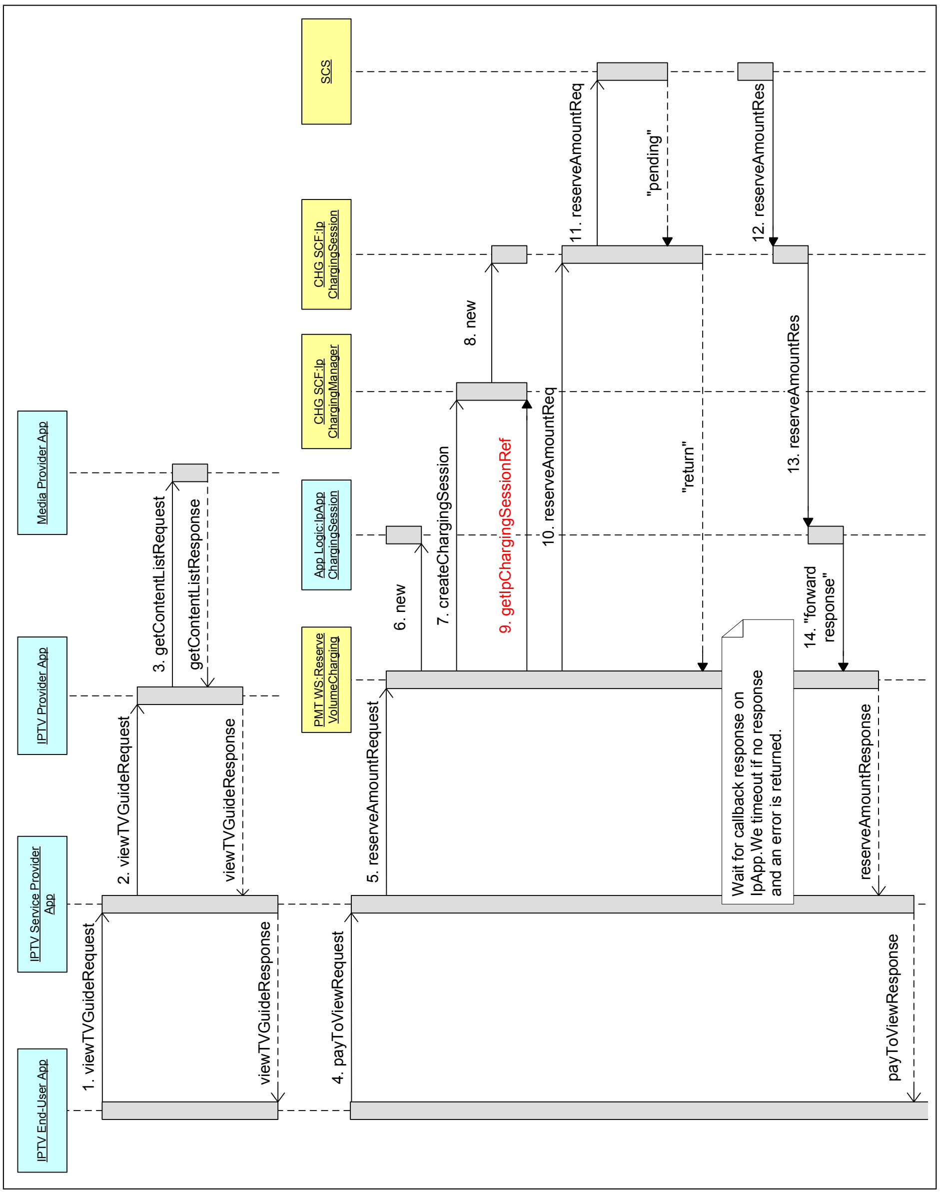


Figure B.11: Pay Per View for Pre-booking Content

11. The SCF requests the SCS to reserve the amount from the end-user's account.

12. The SCS informs the SCF that the amount has been reserved.

13. The SCF informs the web service on its callback object.

14. The callback object informs the web service that the amount is reserved. The web service informs the service provider, who then informs the end-user. Now the delivery of the content is started.

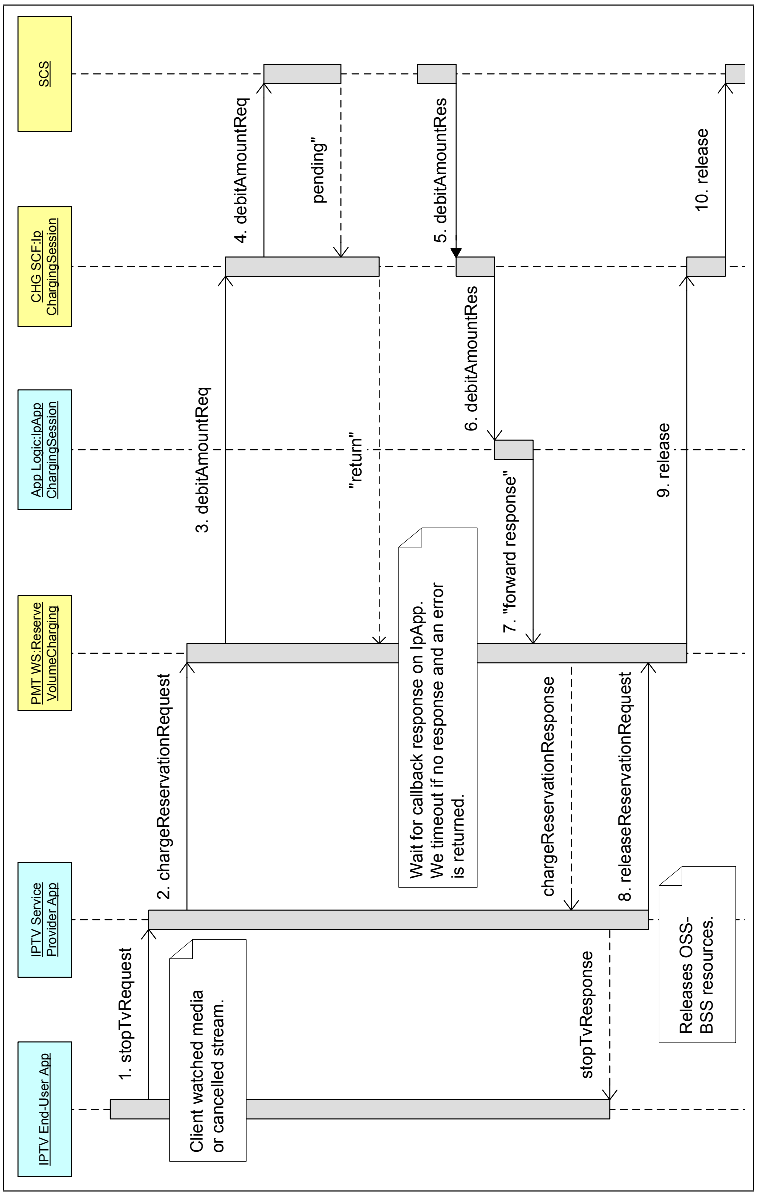


Figure B.12: Pay Per View After Viewing Content

B.1.9 End-User Pay Per View Part 2

In *Figure B.12*, we again use the payment web service API to complete the charging of the end-user account for watching specific content. The figure shows the following interactions:

1. The end-user completes or cancels watching the television programme.
2. The service provider requests the payment web service to complete charging the end-user’s account.
3. The web service requests the payment SCF to debit the reserved amount off the end-user’s account, for watching the specific television content.

4. The SCF requests the SCS to debit the reserved amount off the end-user's account.
5. The SCS informs the SCF that the reserved amount has been debited.
6. The SCF informs the web service on its callback object that the reserved amount has been debited.
7. The callback object informs the web service. The web service informs the service provider, who then informs the end-user.
8. Since the content delivery and the charging session has been completed, the IPTV service provider requests the web service to release all resources used to charge the end-user's account.
9. The web service requests the payment SCF to release all resource used to charge the end-user's account. The web service may also delete its callback object.
10. The SCF requests the SCS to release resources used to charge the end-user's account.

B.1.10 End-User Presence and Availability

In *Figure B.13*, we use the existing Parlay X presence and availability web service API. This web service API simplifies access to the presence and availability SCF. The IPTV application provider uses the presence web service to manage end-user presence and availability for IPTV, telephony and messaging services. The figure shows the following interactions:

1. The end-user updates his/her presence.
2. The IPTV service provider notifies the application provider of the change in end-user presence.
3. The application provider requests the presence web service to store the end-user's presence within the SDP.
4. The web service requests the presence and availability SCF to update the end-user's presence in the SDP.
5. The SCF requests the SCS to update the end-user's presence in a presence database.
6. The end-user requests his/her friend's presence status.
7. The IPTV service provider requests the application provider for the presence status of the end-user's friend.

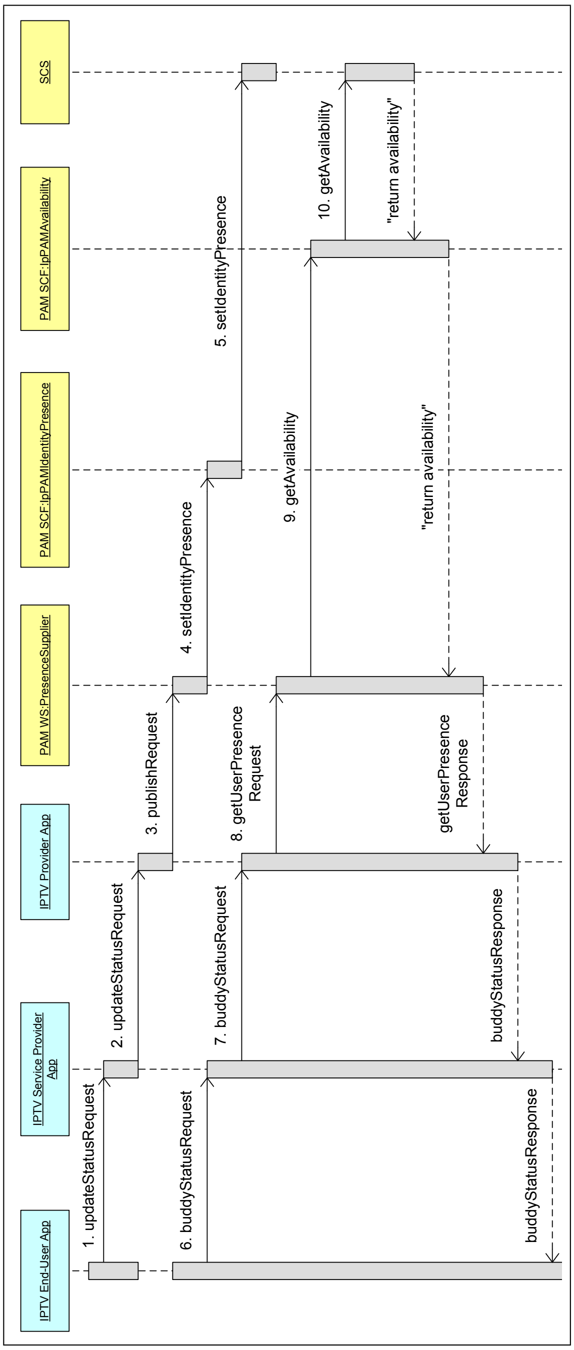


Figure B.13: Update and Obtain End-User Presence Status

8. The application provider requests the web service to obtain the presence status of the end-user's friend.

9. The web service requests the SCF to obtain the presence status of the end-user's friend.

10. The SCF requests the SCS to obtain the presence status of the end-user's friend(s). Once obtained the presence is returned to the end-user, via the SCF, web service, application provider and service provider.

B.1.11 End-User Makes an IPTV Call

In *Figure B.14*, we use the existing Parlay X third party call web service API. This web service simplifies access to the multiparty call control SCF. We do not provide any extensions for this SCF. The IPTV application provider uses the third party call web service to setup and manage calls between end-users registered and using the IPTV service. The figure shows the following interactions:

1. The end-user requests a call be setup with his friend. The friends presence indicates that he/she can answer the call.
2. The IPTV service provider requests the application provider to setup the call.
3. The application provider requests the third party call web service to setup the call between the two parties.
4. The web service creates a call leg callback object for the end-user (Party A).
5. The web service creates a call leg callback object for the end-user's friend (Party B).
6. The web service requests the SCF to create a call object.
7. The call object is created.
8. We add this method to the multiparty call control SCF API, to allow the web service access to the newly created call object's interface.
9. The web service requests the call object to create a call leg object for the end-user (Party A) in the SCF.
10. The call leg object is created.
11. The web service requests the call object to create a call leg object for the end-user's friend (Party B) in the SCF.
12. The call leg object is created.
13. The web service requests notifications from the end-user call leg object about changes in its call state. The web service requests the end-user call leg object to route the call to the end-user across the converged network.
14. The end-user call leg object requests the SCS to route the call in the network to the end-user.
15. The web service requests notifications from the friend call leg object about changes in its call state. The web service requests the friend call leg object to route the call to the friend across the converged network.

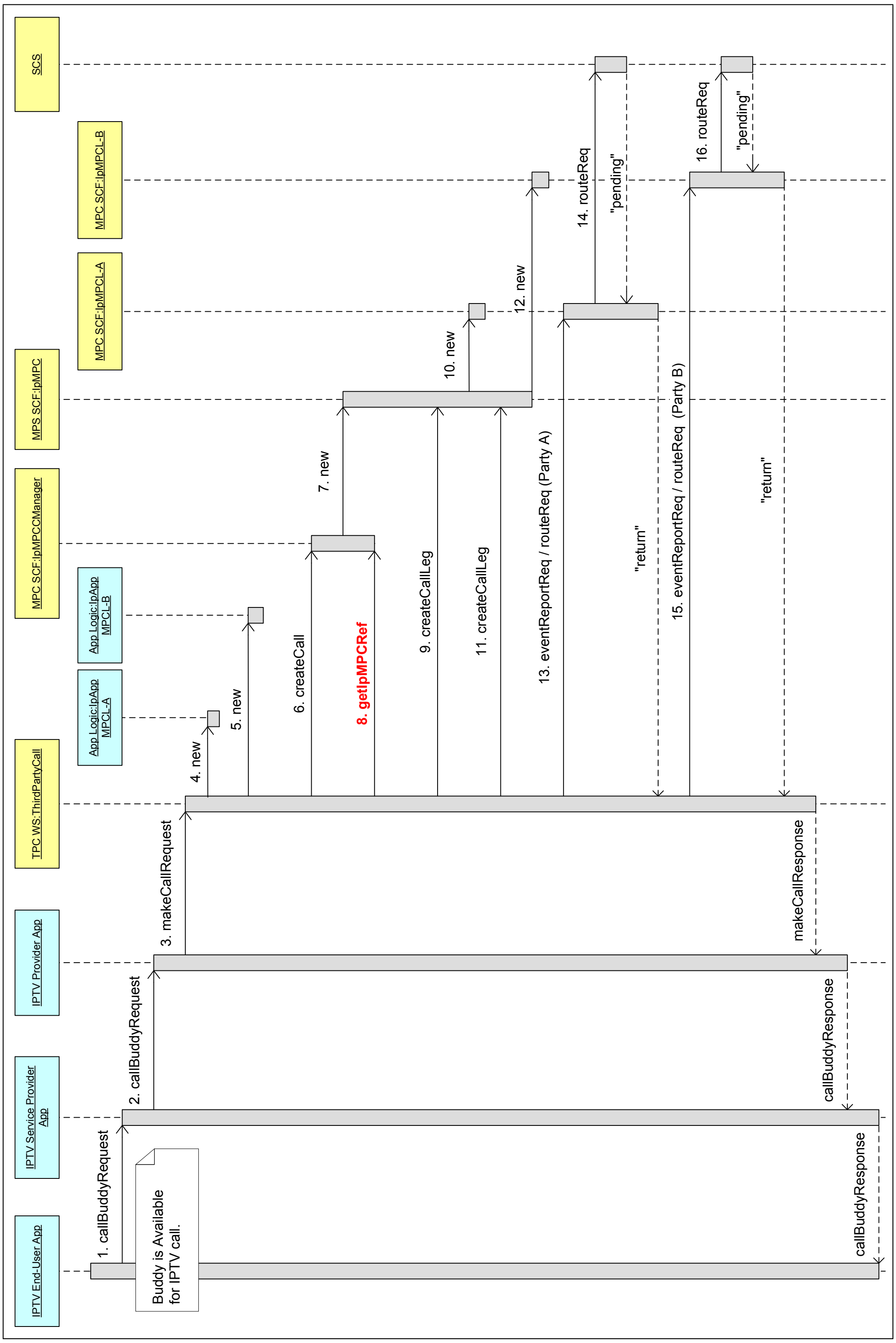


Figure B.14: Setup Two Party Call Between End-User and Friend

16. The friend call leg object requests the SCS to route the call in the network to the friend (Party B). While the call is being setup the web service informs the IPTV application provide that the call is in progress. This information is also returned to the service provider, who also informs the end-user.

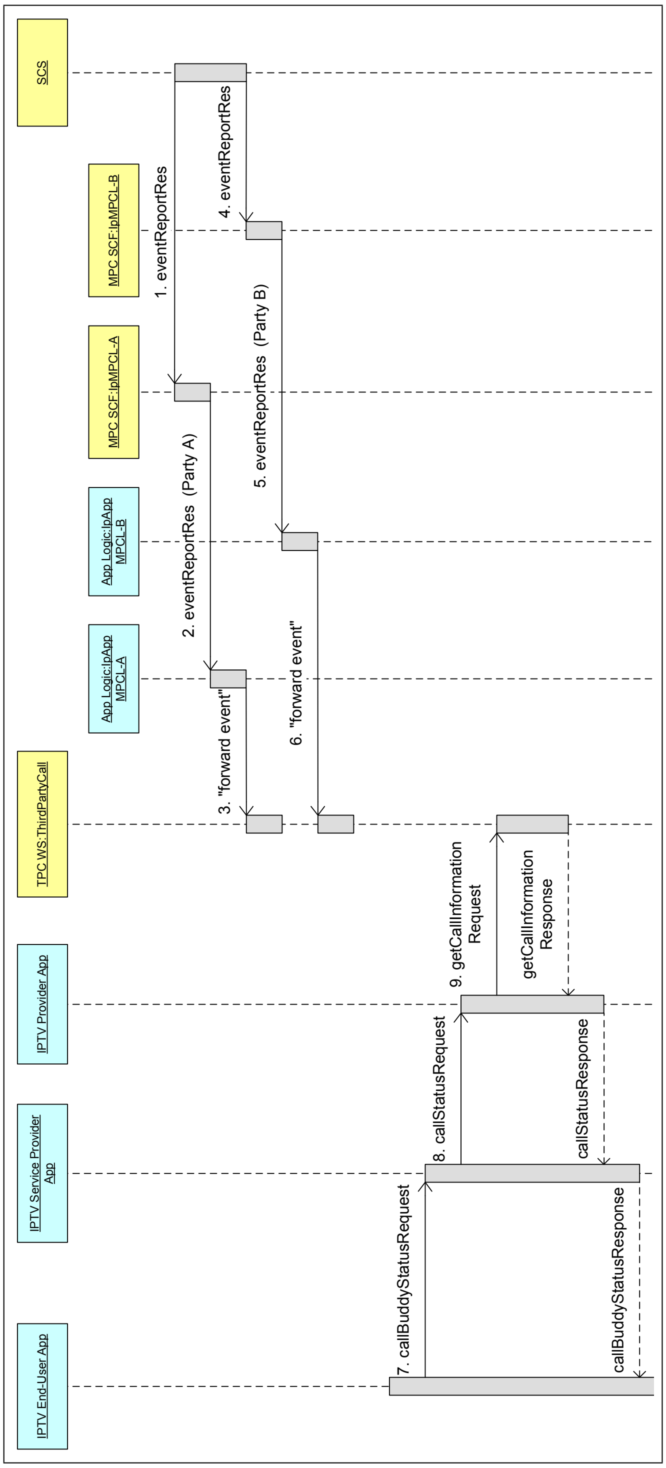


Figure B.15: Checking Status of Two Party IPTV Call

B.1.12 End-User Checks on IPTV Call Status

In *Figure B.15*, we again use the third party call web service API to view the status of call being setup between the two end-users using the IPTV service. The figure shows the following interactions:

1. The SCS informs the SCF's end-user call leg object that the call has been setup between the end-user in the network.
2. The call leg object forwards this result to the web service on its end-user call leg callback object.
3. The end-user call leg callback object forwards this result to the web service.
4. The SCS informs the SCF's friend call leg object that the call has been setup between the friend in the network. Now the end-user and friend are connected to the call and both start communicating.
5. The friend call leg object forwards this result to the web service on its friend call leg callback object.
6. The end-user call leg callback object forwards this result to the web service.
7. At any point in time during the setup of the call, the end-user may request the IPTV service provider for the state of the call.
8. The service provide requests the application provider for the state of the call.
9. The application provider requests the third party call web service for the state of the call. The web service may return waiting, in progress or Party A/B busy as call state information. This information is returned to the application provider. The application provider returns this result to the service provider, who forwards the result to the end-user.

B.1.13 End-User Ends an IPTV Call

In *Figure B.16*, we again use the third party call web service API to end a call that has been setup between two end-users using the IPTV service. The figure shows the following interactions:

1. The end-user requests the call with his friend to be ended.
2. The IPTV service provider requests the application provider to end the call.
3. The application provider requests the third party call web service to end the call between the two parties.
4. The web service deletes its end-user call leg callback object.
5. The web service deletes the friend call leg callback object.

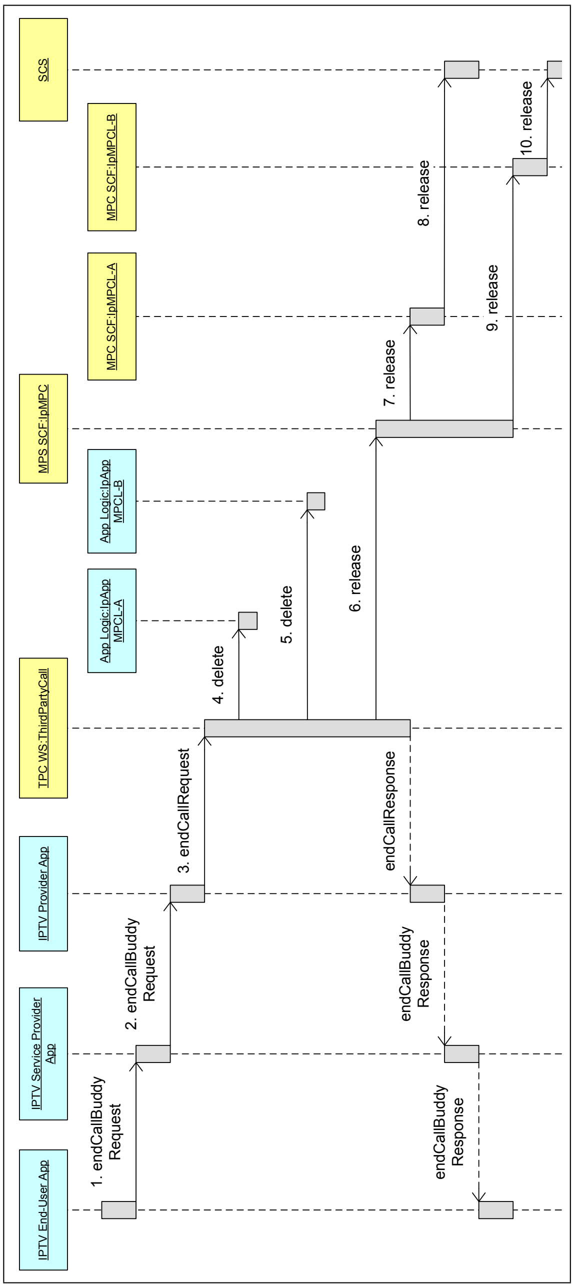


Figure B.16: Ending an IPTV Call between Two Parties

6. The web service requests the multiparty call control SCF release all resources used to setup and maintain the call. The web service returns a successful release result to the application provider. The application provider returns the result to the service provider, who also informs the end-user.

7. The SCF requests its end-user call leg object to release network resources.

8. The end-user call leg object requests the SCS to release network resources used to setup and maintain the call on the end-user side.

9. The SCF requests its friend call leg object to release network resources.

10. The friend call leg object requests the SCS to release network resources used to setup and maintain the call on the end-user's friend side.

B.1.14 Interactive End-User Messaging

In *Figure B.17*, we use the existing Parlay X SMS web service API. This web service API simplifies access to the user interaction SCF. The IPTV application provider uses the SMS web service to send and receive messages between end-users registered and using the IPTV service. The figure shows the following interactions:

1. The end-user requests the IPTV service provider to send a message to his/her friend that is available to read this message.
2. The service provider requests the application provider to deliver this message to the end-user's friend.
3. The application provider requests the SMS web service to send the message to the end-user's friend.
4. The web service creates a callback object that manages asynchronous communication with the user interaction SCF.
5. The web service requests the user interaction SCF to create a new user interaction object in the SCF.
6. The new user interaction object is created.
7. We add this method to the user interaction SCF API, to allow the web service access to the newly created user interaction object's interface.
8. The web service requests the user interaction object to send the message to the end-user's friend.
9. The user interaction object requests the SCS send the message to the end-user's friend. The SCS returns a pending result to the SCF. The SCF forwards this result to the web service. The web service also returns the result to the application provider. The application provider informs the service provider, who also informs the end-user.
10. The end-user's friend may reply to this message after a period of time. The SCS sends the reply message to the user interaction object.
11. The user interaction object informs the web service of this reply on its user interaction callback object.

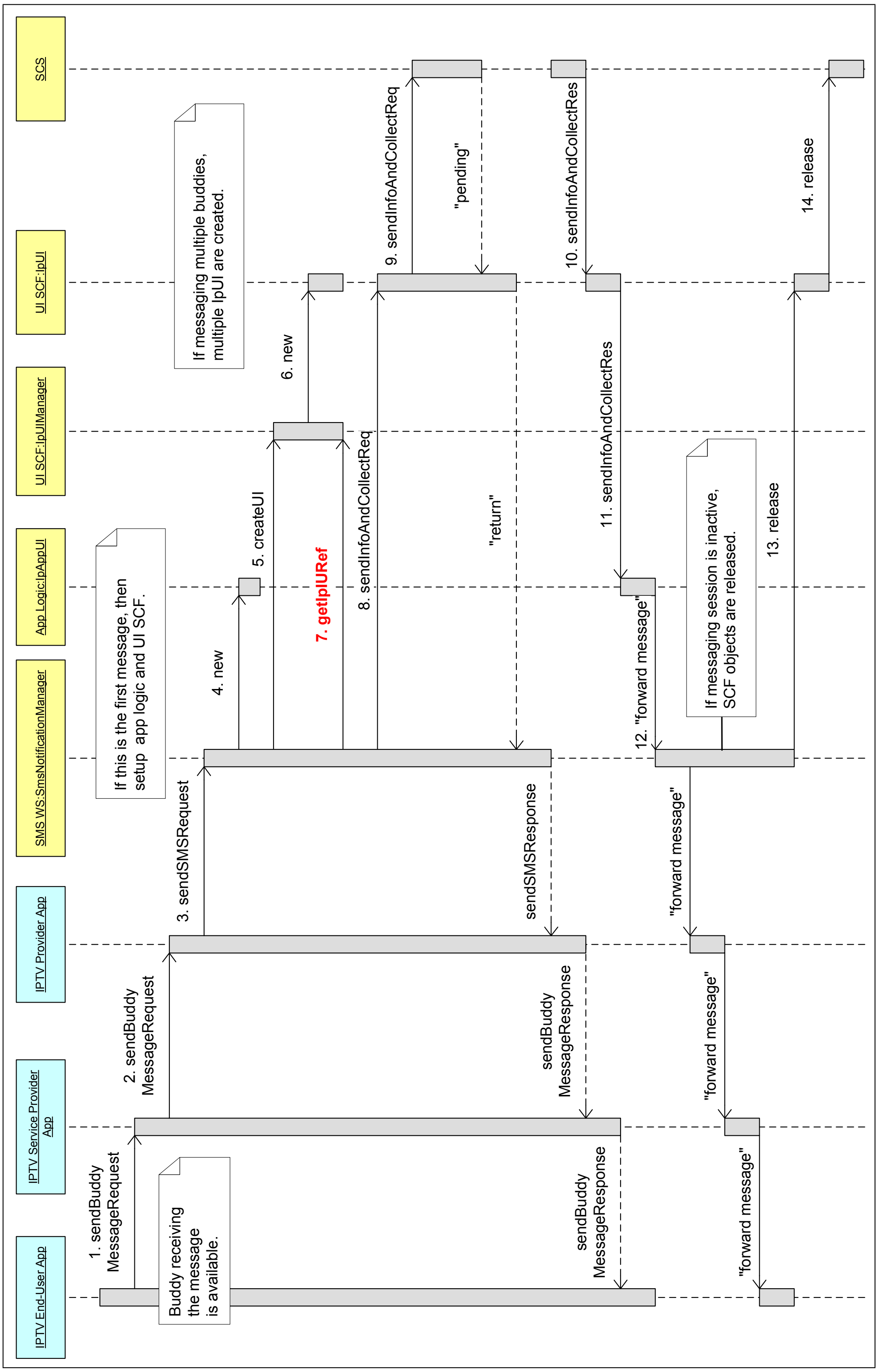


Figure B.17: Messaging between Two IPTV End-Users

12. The user interaction callback object returns the reply message to the web service. The web service forwards the message to the application provider. The application

provider informs the service provider, who also informs the end-user, of the reply message.

13. After a period of time the end-user and friend may not send messages to each other. Hence, the web service requests the SCF's user interaction object to release resources used to facilitate messaging.

14. The user interaction object requests the SCS to also release any resources used to deliver the previous messages.

B.1.15 Provide End-User with IPTV Help

In *Figure B.18*, we use the existing Parlay X audio call web service API. This web service API also simplifies access to the user interaction SCF. The IPTV application provider uses the audio call web service to deliver audio to the end-user, based on interactive responses obtained from the end-user. The pairing of the audio and interactive responses represents the interactive IPTV help service. The figure shows the following interactions:

1. The end-user requests the service provider to start the interactive help service. In parallel a two party call is set up between the end-user and audio source, such as an IVR.

2. The service provider requests the application provider start the interactive help service.

3. The application provider requests the audio call web service to play a specific audio message to the end-user. This initial audio message signals the start of the interactive tutorial.

4. The web service creates a new user interaction callback object to manage asynchronous communication with the SCF.

5. The web service requests the user interaction SCF to create a new user interaction call object.

6. The new user interaction call object is created.

7. We add this method to the user interaction SCF API, to allow the web service access to the newly created user interaction call object's interface.

8. The web service requests the user interaction call object to deliver the specific audio message and collect any input from the end-user.

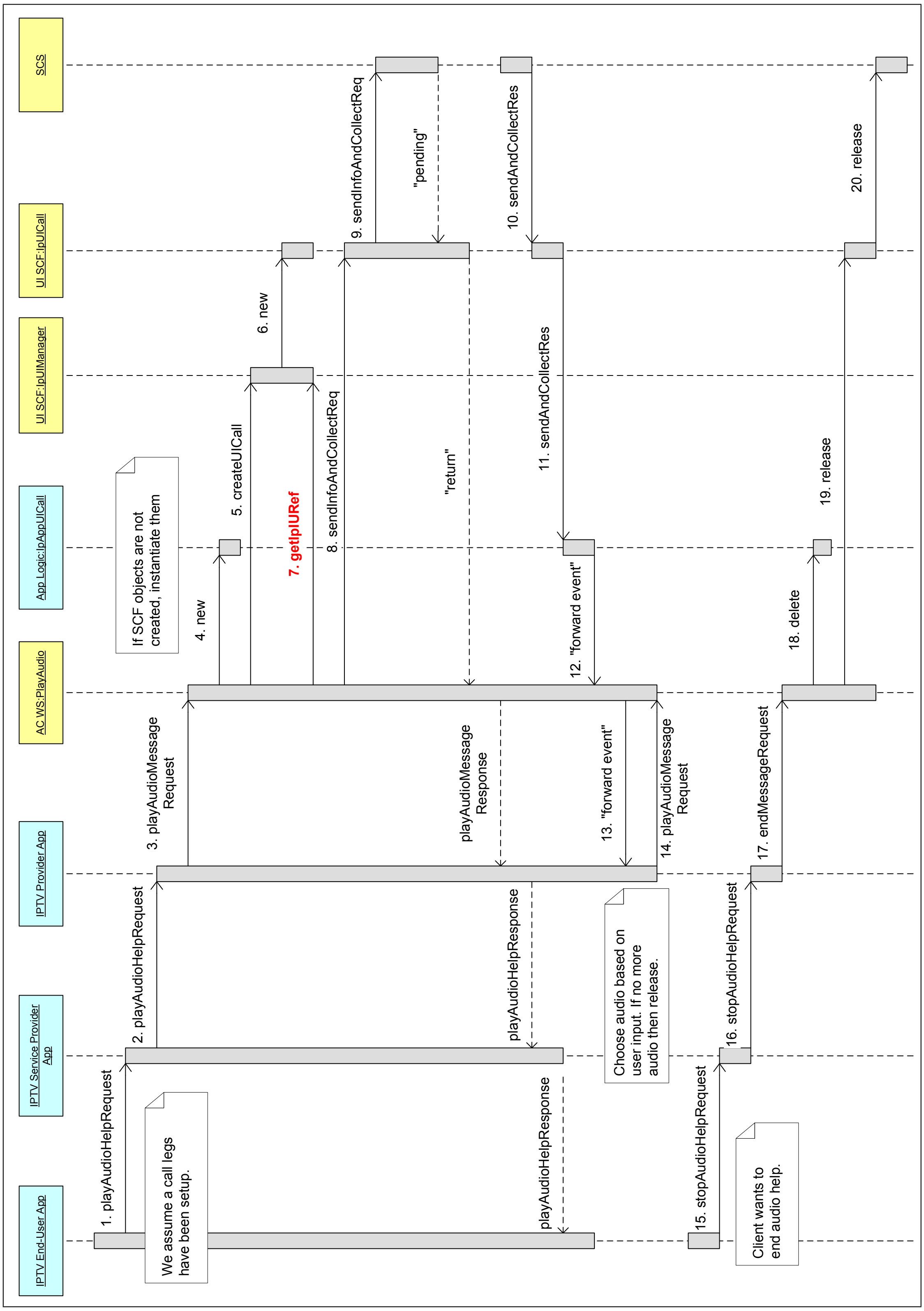


Figure B.18: Using the Interactive IPTV Help

9. The user interaction call object requests the SCS to deliver specific audio from the IVR to the end-user and to collect any input from the end-user. While this may take time to complete, the SCS returns a pending result to the SCF's user interaction call

object. The SCF returns the pending result to the web service. The result is then forwarded, via the application and service provider, to the end-user

10. The SCS returns some end-user input to the SCF's user interaction call object.
11. The user interaction call object forwards this input to the web service's user interaction callback object.
12. The user interaction callback object forwards this end-user input to the web service.
13. The web service forwards this input to the application provider.
14. Based on the end-user input, the application provider determines which audio should be delivered next. The application provider then requests the web service to deliver the audio.
15. The user may request the stoppage of the interactive help service from the IPTV service provider. Also, the two party call between end-user and IVR is ended.
16. The service provider requests the application provider to stop the interactive help service.
17. The application provider requests the audio call web service to stop the delivery of audio to the end-user.
18. The web service deletes its user interaction callback object.
19. The web service requests the SCF's user interaction call object to release resources used to deliver the audio to the end-user.
20. The user interaction call object requests the SCS to release network resources used to deliver the audio to the end-user.

B.2 SCS and IMS Interactions

We defined a SCS API and implementation that interworked with SCS clients to simulate data session, IVR, OSS/BSS and call control network events. The message sequences between SCS and the data session SCS client is shown in *Chapter 10*, *Figure 10.10*. The message sequences between the remaining OSS/BSS, IVR and call control SCS clients and SCS are similar to the data session SCS client.

The SCS client simulators were created since mappings between the SCS API and SIP-based IMS functions are not fully standardised. However, during the design and implementation of the SDP proof of concept we uncovered some SCS to SIP mappings. These

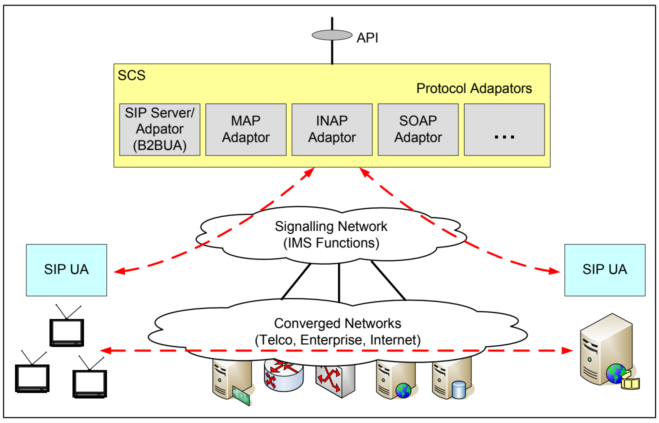


Figure B.19: SCS with Protocol Adaptors

mappings support our SDP implementation and the IPTV service. These mappings are illustrated using message sequence diagrams. Also, these diagrams provide recommendations for SIP message extensions.

B.2.1 Structure of the SCS

In *Figure B.19*, we provide the architectural structure of the SCS. The figure shows the SCS exposing a north-facing interface to SCFs. Inside the SCS are various protocol adaptors. One of these adaptor's is represented as a SIP server. This SIP server is used communicate with the IMS functional entities and the end-user SIP user agents (UA). We also use UAs to represent IMS entities on media sources.

B.2.2 Accessing OSS/BSS and Network Session Capabilities

In *Figure B.20*, we show interactions between the SCS, OSS/BSS, media sources and end-users using SIP. These interactions are initiated by various Parlay SCFs. In the figure we provide recommendations for new SIP messages that allow OSS/BSS functions to be invoked. We reuse existing SIP messages to create data sessions across the converged network. However, we add new SIP messages to manipulate these data sessions.

In *Figure B.20*, we show the following interactions:

1. Account management and charging SCFs request the SCS to invoke various network functions that manage end-user accounts.
2. The SCS invokes the OSS/BSS functions using new SIP messages. These messages

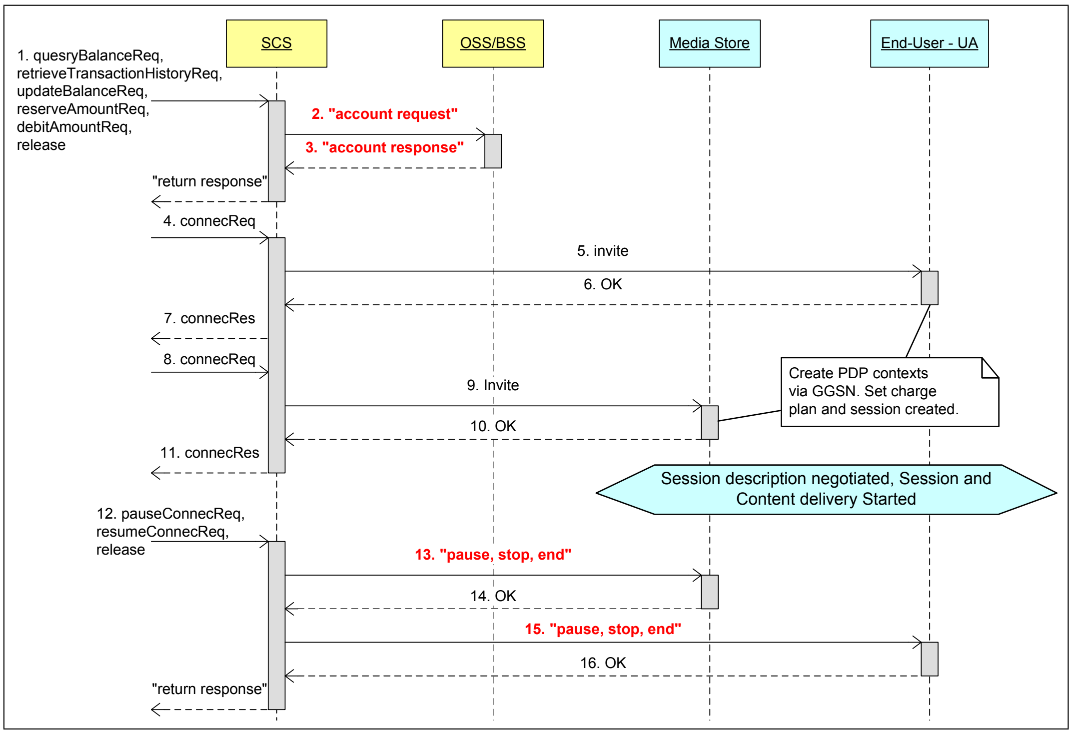


Figure B.20: SCS to SIP mappings for OSS/BSS and Data Sessions

must contain the needed information to invoke the OSS/BSS functions. As an alternative the Diameter protocol may be extended to provide these end-user account management capabilities.

3. The OSS/BSS functions return a result to the SCS using a new SIP message.

4. The data session control SCF requests the SCS to create a network data session with the end-user attached to it.

5. The SCS reuses the SIP Invite message to setup the end-user's half of the data session.

6. The first half of the data session is now created since the SIP OK message is returned from the end-user's UA.

7. The SCS returns a response to the data session control SCF.

8. The data session control SCF requests the SCS to create a network data session with a media source attached to it.

9. The SCS reuses the SIP Invite message to setup the media source's half of the data session.

10. The second half of the data session is now created since the SIP OK message is returned from the media source.

11. The SCS returns a response to the data session control SCF.

12. The data session control SCF requests the SCS to manipulate the existing network data session.

13. The SCS uses new SIP messages to manipulate the end-user's half of the data session.

14. The end-user's half of the data session is altered since the SIP OK message is returned.

15. The SCS uses new SIP messages to manipulate the media source's half of the data session.

16. The media source's half of the data session is now altered since the SIP OK message is returned. Once completed, the SCS returns a response to the data session control SCF.

B.2.3 Accessing Presence and Call Control Network Capabilities

In *Figure B.21*, we show interactions between the SCS, presence server and end-users using SIP. These interactions are initiated by various Parlay SCFs. In the figure we provide recommendations for new SIP messages that allow presence information to be obtained. We reuse existing SIP messages to setup calls between multiple end-users across the converged network.

In *Figure B.21*, we show the following interactions:

1. The presence and availability SCF requests the SCS to update an end-user's presence information on the presence and availability (PAM) server.

2. The SCS uses the existing SIP Publish message to update end-user's presence status on the PAM server.

3. The presence and availability SCF requests the SCS to obtain an end-user's presence information from the PAM server.

4. The SCS uses a new SIP message to request the PAM server for the end-user's presence status.

5. The PAM server uses a new SIP message to return the end-user's presence status. The presence status is returned to the SCF, via the SCS.

6. The multiparty call control SCF requests the SCS to be informed of changes in a call leg being setup between an end-user's (Party A) UA.

7. The multiparty call control SCF requests the SCS to route the call to the end-user's (Party A) UA.

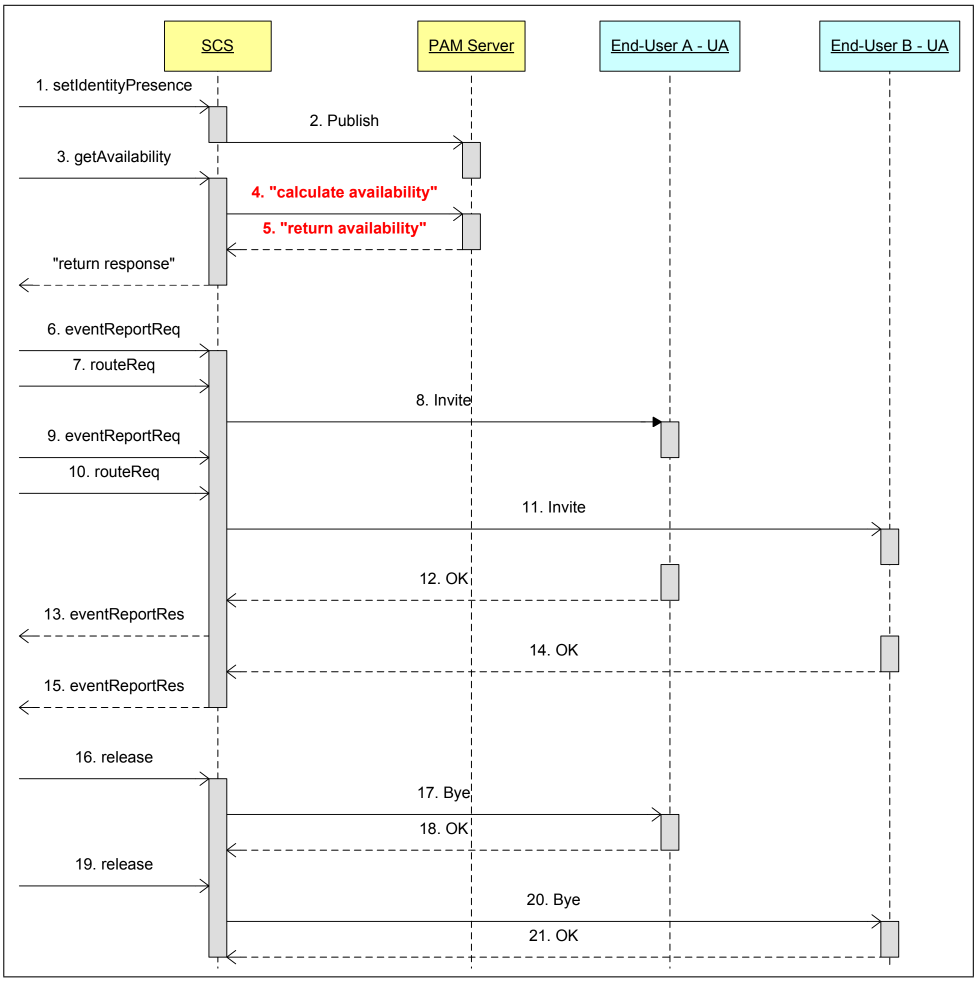


Figure B.21: SCS to SIP mappings for Presence and Call Control

8. The SCS uses the SIP Invite message to setup the call leg involving the end-user's (Party A) UA.
9. The multiparty call control SCF requests the SCS to be informed of changes in a call leg being setup between an end-user's (Party B) UA.
10. The multiparty call control SCF requests the SCS to route the call to the end-user's (Party B) UA.
11. The SCS uses the SIP Invite message to setup the call leg involving the end-user's (Party B) UA.
12. The first end user (Party A) call leg has been setup and the SIP OK message is received by the SCS.
13. The SCS returns a response to the multiparty call control SCF.
14. The second end user (Party B) call leg has been setup and the SIP OK message is received by the SCS.

15. The SCS returns a response to the multiparty call control SCF.
16. Once the call is completed or ended, the multiparty call control SCF requests the SCS to release all resources used to support the call on the first end-user's (Party A) UA.
17. The SCS uses the SIP Bye message to end the call on the first end-user's (Party A) UA.
18. The call is ended on the first end-user's (Party A) UA, since the SIP OK message was received.
19. The multiparty call control SCF requests the SCS to release all resources used to support the call on the second end-user's (Party B) UA.
20. The SCS uses the SIP Bye message to end the call on the second end-user's (Party B) UA.
21. The call is ended on the second end-user's (Party B) UA, since the SIP OK message was received.

B.2.4 Accessing Messaging and Audio Content Network Capabilities

In *Figure B.22*, we show interactions between the SCS, IVR and end-users using SIP. These interactions are initiated by various Parlay SCFs. In the figure we reuse existing SIP messages to enable messages and audio to be delivered end-users across the converged network.

In *Figure B.22*, we show the following interactions:

1. The user interaction SCF requests the SCS to send a message to an end-user (Party B).
2. The SCS uses the SIP Message message to send the message to the end-user (Party B).
3. The user interaction SCF requests the SCS to send another message to another end-user (Party C).
4. The SCS uses the SIP Message message to send the message to the end-user (Party C).
5. In response, the end-user (Party C) uses his/her UA to send a message using the SIP Message. This message is returned to the user interaction SCF via the SCS.

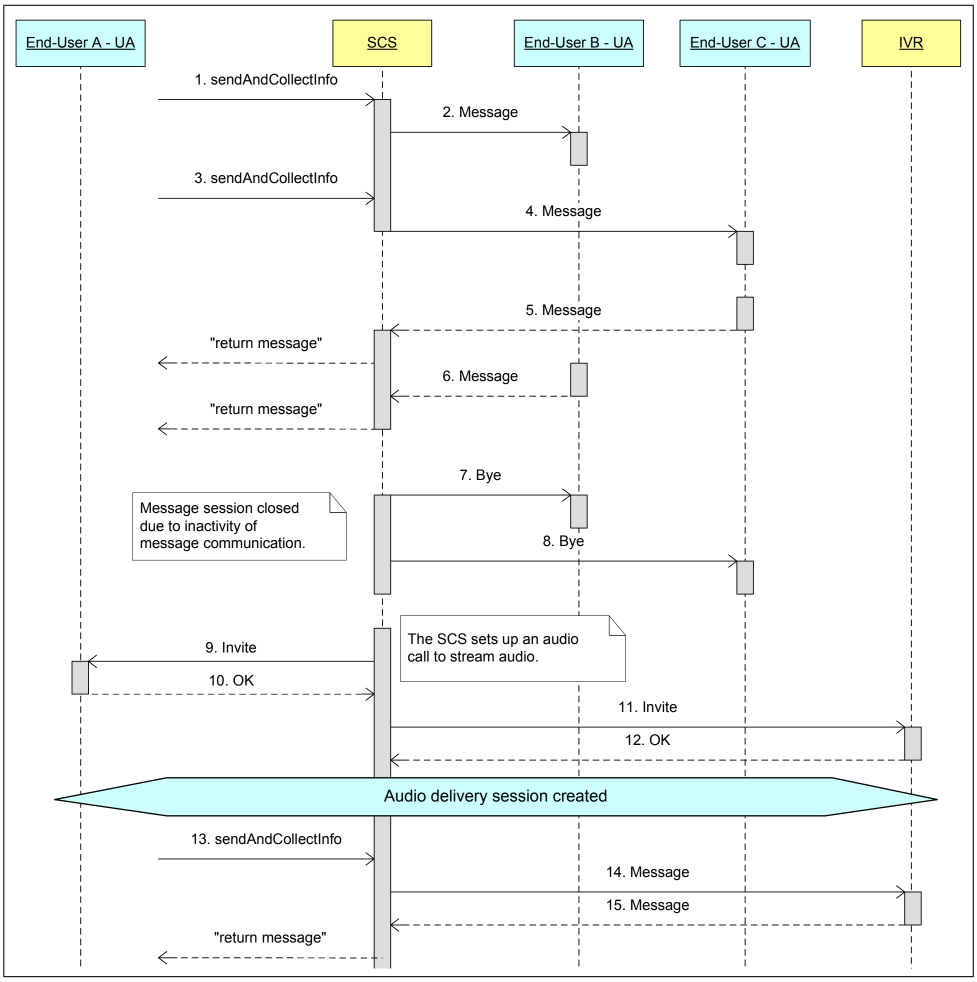


Figure B.22: SCS to SIP mappings for Messaging and Interactive Audio Delivery

6. After some time, the end-user (Party B) also uses his/her UA to send a message using the SIP Message. This message is returned to the user interaction SCF via the SCS.

7. Since no more messages are being sent, the SCS releases resources used to deliver messages to end-user (Party B). The SCS reuses the SIP Bye message.

8. Since no more messages are being sent, the SCS releases resources used to deliver messages to end-user (Party C). The SCS reuses the SIP Bye message.

9. During the activation of the IPTV interactive help service, the multiparty call control SCF requests the SCS to setup an audio call between an end-user (Party A) and IVR. The SIP Invite message is used to setup the audio call between the end-user.

10. The audio call is setup since the SIP OK message is received by the SCS from the end-user (Party A).

11. The multiparty call control SCF requests the SCS to complete the setting up of the audio call. The SIP Invite message is used to setup the call between the IVR.

12. The audio call is setup since the SIP OK message is received by the SCS from the IVR. Now the appropriate audio can be delivered to the end-user.

13. The user interaction SCF requests the SCS to play a specific audio file that is heard by the end-user (Party A).

14. The SCS uses the SIP Message message to inform the IVR to play a specific audio file, that is heard by the end-user.

15. As a result of hearing the audio, the end-user inputs some key on his/her terminal. The key generates a audio tone that is captured by the IVR across the audio session. The IVR returns this input using the SIP Message message to the SCS. The SCS forwards this end-user input to the SCF.

Appendix C

Lessons Learned from the SDP Proof of Concept

C.1 Benefits of an Integrated Development Environment

The Java programming language and tools supported the quick development of the SDP proof of concept. In addition, we used a popular integrated development environments (IDE) to support Java development. We found the Netbeans IDE to be effective and efficient when creating the Java-based SDP proof of concept. This IDE simplified development tasks and provided Java-oriented developer tools and support. Also, the IDE helped with web service development, but not deployment (discussed later).

C.2 Richness of CORBA

The standard-based CORBA middleware abstracted various distribution complexities in our SDP proof of concept. It supported communication across the distributed SDP services, including the SCFs, SCS and SCS clients. Also, CORBA provided various middleware services to support the distributed SDP services. One vital middleware service was the naming service that enabled distributed applications and SDP services to easily locate each other. However, on deployment each distributed application and service had to be provided with the location information of the naming service. This eased redeployment of components across different servers and network areas.

CORBA provided another middleware service called the portable interceptor service. The portable interceptor service is part of the CORBA middleware. This service's capabilities were inherited by our own SDP interceptor service, so as to intercept requests made on SCF, SCS and SCS client APIs. As a result, we were able to verify if the correct methods were being invoked with the correct parameter values on the SDP service APIs. With the help of the CORBA interceptor service, our SDP interceptor service also intercepted the corresponding replies from SCS clients, SCS and the SCFs. This enabled us to verify if the correct information was being returned by the SDP services. Hence, the interceptor service contributed to the testing of our SDP's distributed services.

In *Figure C.1* we illustrate another example of using the portable interceptor service within the SDP. This example is generic and uses the SDP interceptor service to uphold telco policies and log SCF and SCS usage information in the telco OSS/BSS. In the figure we show the following interactions:

1. The SDP interceptor service registers with the CORBA portable interceptor service.
2. The SDP interceptor service registers for notifications of requests on specific SCFs.
3. The SDP interceptor service registers for notifications of replies made by specific SCFs.
4. An application invokes a request on a web service.
5. The web service makes a request on a specific SCF. However, this request is intercepted by the CORBA portable interceptor service.
6. The CORBA portable interceptor service determines that the SDP interceptor service requires notification of these SCF requests. A notification is sent.
7. The SDP interceptor service requests the Policy SCF to decide whether the application and web service are allowed to use the current SCF.
8. The Policy SCF informs the SDP interceptor service that the SCF request is allowed to be used by the application and web service.
9. The SDP interceptor service informs the CORBA portable interceptor service to allow the the request to continue.
10. The CORBA portable interceptor service forwards the request to the appropriate SCF.
11. The SCF invokes the SCS.
12. The SCS returns a result to the SCF.
13. The SCF perceives the result to be returned to the web service. However, the CORBA portable interceptor service intercepts the response to the web service.

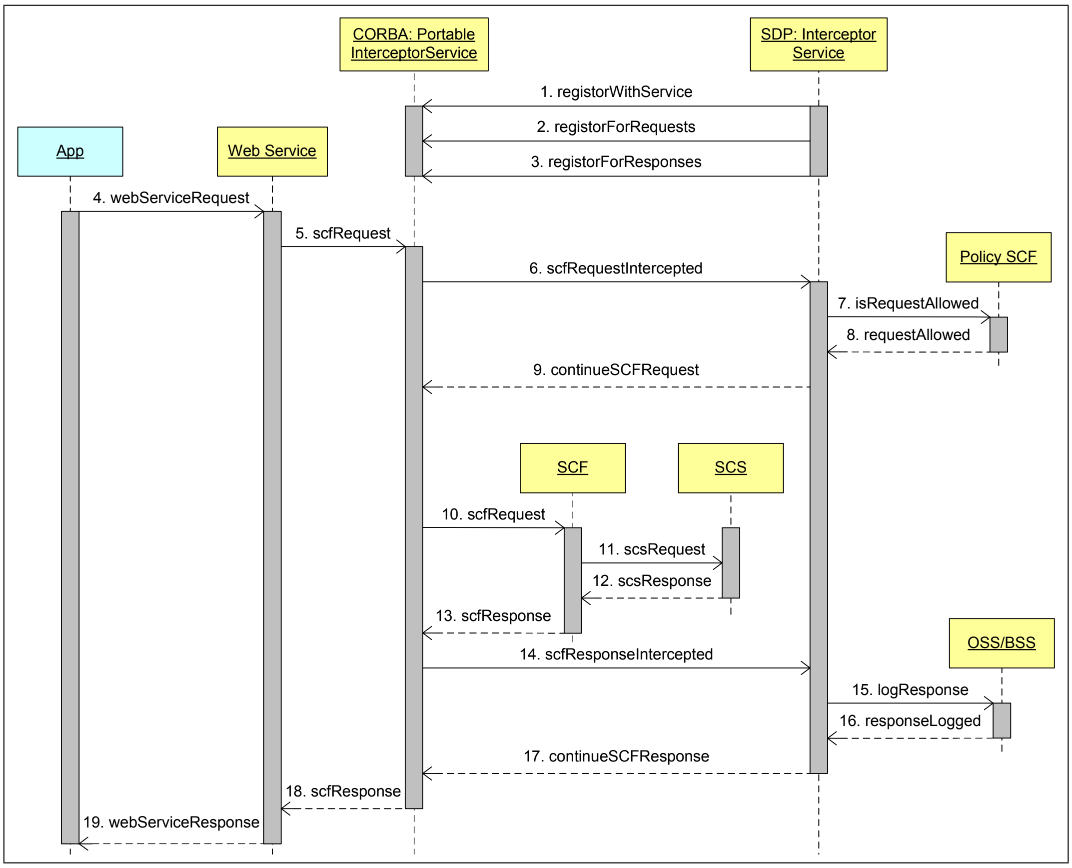


Figure C.1: Using CORBA Portable Interceptors

14. The CORBA portable interceptor service determines that the SDP interceptor service requires notifications of these SCF replies. A notification is sent.

15. The SDP interceptor service requests the OSS/BSS to log the information contained in the SCF response.

16. The OSS/BSS successfully logs the response information and informs the SDP interceptor service of this result.

17. The SDP interceptor service informs the CORBA portable interceptor service to allow the response to be delivered.

18. The CORBA portable interceptor service forwards the response to the web service.

19. The web service returns a response to the application.

C.3 Problems with Web Services

In contrast to the CORBA-based SCFs, SCS and simulators, we faced many problems with the web service programming, deployment, testing and ESB middleware. Some of these problems are as follows:

- Web service interfaces (WSDL) did not provide a complete abstraction of their distribution and location. This increased the complexity of redeploying web services on different application servers across the network. To overcome this problem we used the Netbeans IDE to copy the web service project to a portable storage drive. We then used Netbeans to copy the project from the portable drive to a location on the target server. However, the various inconsistencies in the web service's code had to be corrected to reflect the changes in its distribution.
- The stateless property of web services provided a challenge. For example, web services had to continuously poll SCFs for call status and end-user presence information. There is no solution for this problem since we could not find standards-based solutions for asynchronous web services.
- The ESB middleware bundled in the application servers provided proprietary functionality. This included middleware services that exposed functions to Java-specific mechanisms, such as Java message queues. Hence, our web service implementations could not operate across any other ESB middleware.
- Each application server required detailed knowledge of the network and its configuration, such as proxy and host list. This information must detail elements that can be reached with or without the use of proxies.
- Due to configurations the application servers required redeployment of web services after each simulation. It proved too time consuming to reconfigure the server, since the documentation and configuration files are complex to understand and use.

Therefore, web service middleware is complex and provided limited standard-based functionality. Also, the middleware did not provide additional benefits to our web service implementations and SDP proof of concept.

Appendix D

Source Code

The implementation of the SDP framework can be found on the accompanying compact disc. The disc contains a README.txt file describing in detail the contents of the disc.