J Internet Serv Appl (2011) 2:95–111
DOI 10.1007/s13174-011-0028-9

SI: FUTURE NET SERVICE MODELS & DESIGNS

From service delivery to integrated SOA based application delivery in the telecommunication industry

Christian Menkens · Michael Wuertinger

Received: 31 January 2011 / Accepted: 18 May 2011 / Published online: 19 June 2011

Abstract Currently, in the Telecommunication industry, the move toward service oriented infrastructures based on Internet Protocol (IP), IP Multimedia Subsystem (IMS), Service Oriented Architectures (SOA), and Service Delivery Platforms (SDP) can be observed. Communication Service Providers (CSP) are opening parts of their core infrastructure and interfaces to third party developers. Nevertheless, there is still a gap between application developer communities and the Telecommunication industry. CSPs offer their services so developers can use them in their applications but developers do not have any incentives, tools, or development environments that would help them to overcome the barriers of accessing Telecommunication services to build future distributed Web-Telecom-Converged Applications. This research work presents a concept for how to close this gap by conceptualizing an end-to-end platform from Telecommunication core networks and the Internet to end user clients. Requirements of all platform stakeholders are identified and through the combination of SOA platform concepts with Telecommunication SDP architecture concepts and application store application distribution concepts (e.g. Apple AppStore, Google Market) fulfilled. The final result is a concept and logical reference architecture for Application Delivery Platforms (ADP) that extend the concepts of SDPs by broadening the focus from re-usable SOA services over a Web-Telecom-Converged Application architecture and service/application life cycle management to end user applications on client platforms and devices. In order to proof the concepts of the logical ADP reference architecture, central parts of the architecture and a demonstrational "Converged Address Book" use-case are implemented and deployed.

Keywords Service delivery platform · Service oriented architecture · Telecom web convergence · IP multimedia subsystem

C. Menkens (✉) · M. Wuertinger
Center for Digital Technology and Management, Technische Universitaet Muenchen, Munich, Bavaria 80290, Germany
e-mail: menkens@cdtm.de

1 Introduction

Currently, in the Telecommunication industry, the move toward service oriented infrastructures based on Internet Protocol (IP) [29], IP Multimedia Subsystem (IMS) [7], Service Oriented Architecture (SOA) [38, 39] using for example ParlayX [12], and Service Delivery Platforms (SDP) [24, 25, 47–49] can be observed. In addition to that, Communication Service Providers (CSP) are opening parts of their core infrastructures and interfaces to third party developers (3rd Party Enabling [49]). For the first time, it is possible for developers to converge Internet based Web2.0 [68] services and applications provided by Web/Internet Application Service Providers (ASP) with Telecommunication services and applications provided by CSPs in a distributed Web-Telecom-Converged Application accessible through various different end user clients as depicted in Fig. 1.

These transformations in the Telecommunication industry combined with the rise of programmable and open operating systems for computers, (IP)TVs [72], and mobile devices such as Linux [71], Apple iPhoneOS [70], Google Android [73], Symbian [69], MeeGo [74], Windows Phone 7 [75], etc. lead to new possibilities for application visionaries and application developers. This provides an environment for the development of distributed and ubiquitous applications and services that has never been available previously.

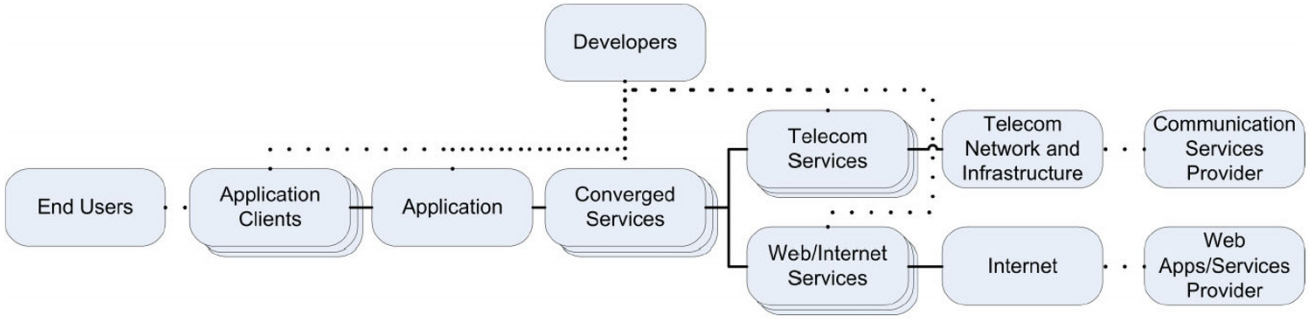

Fig. 1 Distributed Web-Telecom-Converged application and services

2 Problem and motivation

In the Telecommunication industry, research and specification efforts are focused on Enablers [23], Service Environments [24], and SDPs around the Telecommunication core infrastructure. The most widely used and referenced specifications in this area are the Open Mobile Alliance (OMA) Service Environment (SE) [24] and OMA Service Provider Environment (OSPE) [25]. These specifications define how Telecommunication functionalities and features can be encapsulated into services and exposed to Telecommunication external developers. Unfortunately they stop at the exposure of the Telecommunication services and do not provide any concepts for composing Telecommunication services with Web/Internet services or supporting developers at integrating the exposed services into their end user applications. Further, service exposure specifications such as ParlayX [12] or OneAPI [93] focus only on business information system technologies for service exposure such as Simple Object Access Protocol (SOAP) [19] and are not compatible with today's common Web/Internet technologies such as Hypertext Transfer Protocol (HTTP) [32], Hypertext Markup Language (HTML) [21], Asynchronous JavaScript and XML (AJAX) [90], and Representational State Transfer (REST) [33], etc.

Telecommunication industry specifications and standards dealing with end user clients like computers, (IP)TVs and mobile devices are limited to provisioning and configuration specifications such as OMA Client Provisioning [26] or OMA Browsing [27]. These mainly focus on network protocols and connection/communication configuration of client devices and not on application development and application life cycle management. They do not take into account the Open Systems Interconnection (OSI) Application Layer [28] that spans from basic Web/Internet or Telecommunication service usage in applications to clients on end user devices and platforms.

Modern and widely used mobile application development platforms such as iPhoneOS, Android, or Symbian do not support Telecommunication industry specifications such as IP Multimedia Subsystem (IMS) [7] and/or Session Initiation Protocol (SIP) [31] by default and mobile application developers need to use external libraries such as PJSIP [77] or MJSIP [78] to develop applications that use services from the Telecommunication industry. Due to that, in the application development industry developers, applications and research are oriented toward Internet standards, specifications and protocols such as IP, HTTP, JavaScript Simple Object Notation (JSON) [79], Extensible Markup Language Remote Procedure Call (XML-RPC) [80], REST, etc.

The development communities around these Internet technologies oriented platforms are large and successful. Communities such as Apple's with around 28,000 developers developed around 134.000 mobile applications for the iPhone [81] and Google's with around 100,000 developers [82] develop thousands of Web/Internet applications on the Internet as well as mobile applications for the Android platform.

These trends show that there is a gap between application developer communities and the Telecommunication industry. The Telecommunication industry offers its services so developers can use them in their applications. Developers do not have any incentives, tools, or development environments that would help them to overcome the barriers of accessing the Telecommunication services and to build distributed Web-Telecom-Converged Applications. As a result, application developers stick to the easier accessible Internet standards, specifications and protocols, and develop over-the-top applications that take the Telecommunication networks as granted foundation and use them as bit pipes for their application data. This leads to a re-implementation and re-invention of existing Telecommunication services based on Internet technologies. This is problematic since existing Telecommunication services have a much higher level of quality and reliability [84] than the Internet technologies based re-implementations and the existing ones should instead be re-used from CSPs in future applications. This also leads to proprietary/incompatible solutions for these re-implemented and re-invented services (e.g. Voice Call over Skype [83]), limits the possibilities developers have for new

applications and prevents the forthcoming of Web-Telecom-Converged Applications.

In terms of developer communities, this leads to closed and proprietary models for application stores [88, 89] and developer communities that are heavily tied to one hardware (e.g. iPhone) or one platform whereas, in order to build Web-Telecom-Converged Application across all screens (TV, Mobile, PC, ...) a user uses, open and compatible concepts for application delivery and developer communities have to be aimed for.

3 Contributions

This research work presents a concept for how to close this gap between the Telecommunication industry and developers by conceptualizing an end-to-end platform from Telecommunication core networks and the Internet to end user clients. Requirements of all platform stakeholders are identified and through the combination of SOA platform concepts with Telecommunication SDP architecture concepts and application store application distribution concepts (e.g. Apple AppStore, Google Market) fulfilled. The final result is a concept and logical reference architecture for Application Delivery Platforms (ADP) that extend the concepts of SDPs by broadening the focus from re-usable SOA services over Web-Telecom-Converged Application architecture and service/application life cycle management to end user applications on a wide variety of client platforms and devices.

At first, the work presents an overview about the different requirements of the platform stakeholders "Communication Service Providers", "Application Developers", "End Users" in this Web-Telecom-Converged Applications ecosystem. Then it outlines a concept for logical ADP reference architecture that extends the concept of SDPs. Finally, to proof the concepts of the logical ADP reference architecture, central parts of the logical ADP reference architecture and a demonstrational "Converged Address Book" use-case are implemented and deployed on.

4 Related work

4.1 IP Multimedia Subsystem (IMS)

The IP Multimedia Subsystem (IMS) [7] standard defines a set of specifications for offering mobile and fixed Voice over IP (VoIP) and multimedia services. The IMS standard was introduced by the Third Generation Partnership Project (3GPP) [10] as a part of their Release 5. It supports multiple access types, including Global System for Mobile Communications (GSM) [1], GPRS [8], EDGE [8], Universal Mobile Telecommunication System (UMTS) [9], Wireless Local Area Network (WLAN) [2], and wire line broadband (e.g. DSL [3]) access. IMS truly merges the Internet with the cellular and fixed Telecommunication world and uses standardized Telecommunication technologies to provide ubiquitous network access and Internet technologies to provide appealing services. It enables the integration of IMS specific enabling services, such as Presence [35], Location [11], Picture/Text Messaging [34], Video/Voice [4], etc. into one application and supports Charging [5] and Quality of Service (QoS) [6]. IMS shortens application development time and supports deployment and re-use of applications and services. IMS uses IP as its underlying network protocol and SIP, a text-based, by developers simply comprehensible and built on a straightforward request-response interaction model protocol, as signaling protocol. This enables IMS applications to interact with any current and future Internet service and applications by defining and using standard interfaces over an IP-based infrastructure.

4.2 Service Delivery Platforms (SDP)

In general, SDPs define the use of Telecommunication systems and Web/Internet technologies to provide CSPs with a service delivery architecture that supports service creation, session control, standardized protocols and service exposure. SDPs enable CSPs to change their business models by opening up service infrastructures to third parties and support rapid creation of new types of services and applications. SDPs are an enabler for innovation inside and outside the CSPs service and network infrastructure [87].

SDPs have grown steadily in importance over the last five years [87] and their roots can be traced back to mobile content management and delivery systems. These were approaches to replace proprietary Intelligent Network (IN) platforms with standards based Java environments and more recent efforts to open service infrastructures using technologies such as W3C Web Services [15] (e.g. ParlayX [12]). Current SDPs provide an environment in which software developers can efficiently develop new applications by combining the capabilities of existing Telecommunication services (e.g. IMS Voice/Video) and/or service enablers (e.g. IMS Messaging or Presence Enablers) and enable efficient collaboration with content providers and third-party service providers.

There is no standard definition for SDPs but the TM Forum, Alcatel-Lucent and OMA published the Service Delivery Framework (SDF) [48], the Service Delivery Environment [47] and the OMA Service Provider Environment (OSPE) [25] specifications which are central references in the area of SDPs in the Telecommunication industry. The OSPE is the currently most widely used specification in research and development projects in the Telecommunications domain and is used as a basic foundation for the Telecommunication SDP concepts of this research work.

4.3 Web-Telecom-Convergence services and applications architectures

A SDP typically delivers services within only one domain such as the Telecommunications domain which applies to, for example, the SDPs described in [24, 25, 47, 48].

A Web-Telecom-Convergence architecture goes beyond the one of an SDP by bridging the gap between the Telecommunication and the Web/Internet domain. A number of research projects propose different architectures for Web-Telecom-Convergence on different levels of abstraction with different scopes.

Blum et al. propose in [43, 44] a Service Delivery Framework based on SOA concepts and present a prototype implementation and testbed deployment. Their platform allows for an autonomous composition of Telecommunication services within the Telecommunication domain using SOA service description and orchestration concepts. Web-Telecom-Convergence of services and applications is out of scope of their platform and defined to take place on layers above their Telecommunication services layer.

Blum et al. present in [45] a concept for a policy based service broker that allows for an integration of Telecommunication services based on IMS enablers into Web/Internet Mashups and other third party applications. The broker mediates between the real-time Telecommunication services and third party applications and the broker's policies serve as service access definitions as well as an expression for user/service specific capabilities and rules.

Deinert et al. propose in [41] an approach to combine Web2.0 enabled widgets with Telecommunication services exposed by an IMS. They introduce a new abstraction layer with interfaces and a widget engine that provides access to the Telecommunication services. It allows rapid development of Web2.0 widget applications but is restricted to use cases where widgets are suitable as end user clients.

Yang and Park propose in [50] an Open Service Access Gateway (OSAG) that allows Telecommunication operators to leverage W3C Web Services to provide integrated services. It further describes how Telecommunication and W3C Web Services can be combined on the OSAG to build Web-Telecom-Converged Services and how applications can use these services through defined interfaces. An application architecture or end user client support is out of scope of this research work.

Gbaguidi et al. propose in [51] the concept of Hybrid Services as services that span across multiple, especially IP-based, networks such as telephone, Internet, and cellular. They describe an architecture for the provisioning of such services. Their architecture allow programming elements of the platform in order to develop new services to be used in applications. Further, they introduce a concept of orchestration on Telecommunication features such as call control, security, address translation, etc. and expose them as higher level services for re-use. Finally, they implement a Java based prototype of the platform and a prototype service/application on top.

Shao et al. describe in [52] an integrated Telecommunication and IT Service Delivery Platform for converging IMS Services, W3C Web Service and underlying services. They propose a combination of their SIP-based Micro Service Orchestration with a Service Bus to achieve the orchestration. At the end, they present a prototype multimedia service implementation and discuss the support of their Service Delivery Platform for rapid development and deployment of new Web-Telecom-Converged Multimedia Services.

Ohnishi et al. present in [49] a SDP concept that goes beyond the ones discussed in Sect. 4.2. Their goal was to develop an SDP architecture that co-exists with Enterprise and Internet architectures, and so they use Enterprise/Internet compatible Web/Internet and W3C Web Service technologies such as ParlayX [12], Business Process Execution Language (BPEL) [96], Enterprise Service Bus (ESB) [97] and Web Service Description Language (WSDL) [18] to realize their SDP. They extend the technologies used in order to fulfill Telecommunication service specific requirements identified from a network service carriers viewpoint.

The most relevant related works presented above in addition to [23–25, 46, 53–67] build the foundation of this research work. Their analysis showed that a concept of a complete end-to-end SOA based reference architecture of an ADP for Web-Telecom-Converged Applications and services that fulfills all the requirements, identified in the next section, has not been proposed.

5 Requirements of stakeholders

In the area of ADPs, a triangle between the different requirements of the three stakeholders "Communication Service Providers", "Application Developers", and "End Users" exists. In an ADP concept, these requirements need to be considered and balanced so the resulting ADP adds value for all stakeholders.

5.1 Communication Service Providers (CSP)

In the end user applications ecosystem, CSPs compete with over-the-top application providers such as Google and Apple. This leads to the trend that an increasing number of end user applications are provided by these over-the-top providers. CSPs are increasingly pushed into a position where they only provide the technical infrastructure and networks for data transmission ("becoming a bit pipe") which they try and need to anticipate. In order to be successful in this competition, CSPs need to increase their visibility

in the application developer space and present themselves as high quality service providers to developers. One way to do that, which several CSPs already follow, is exporting/offering Telecommunication services to developers on developer platforms and offering support, tools, and business/revenue/marketing models to attract developers (e.g. Developer Garden [36], BetaVine [85], Orange [86], etc.). Since Telecommunication services implement high quality standards [84], there is a need for concepts and means for how to ensure quality of applications developed by external developers before they are offered and distributed to end users [87].

In addition to exporting Telecommunication services, CSPs also need means to combine existing Web/Internet based services with internal Telecommunication services in order to offer value added services to external developers and applications to their customers. Mostly Web/Internet services do not comply with the high quality standards of Telecommunication services, so there need to be concepts for dealing with quality problems of imported services that help to keep the high quality user experience of end users in tact even if there are failures.

5.2 Application developers

Developers want development platforms that allow them to use the latest platform/hardware features and Web/Internet/Telecommunication services available, to develop their applications quickly and to distribute their applications to end users easily. For the use of latest features and quick development, the development platform needs to have a low barrier of entry. This includes that developers have to be able to contribute with minimal costs and minimal hardware requirements, development tools need to be provided, the platform needs to use state-of-the-art and well known programming languages, SDKs and API concepts. Those have to be easy to learn and they need to have as much access to the client platform as necessary for their innovative application ideas [87].

For easy distribution of applications, developers require a defined and working distribution channel to as many potential end users as possible. This channel needs to allow and assist them at promoting, selling, and distributing their applications on end user devices worldwide.

5.3 End users

End users require innovative and easy to use applications that support them at their work or in their daily life. For the end user, it needs to be easy and user friendly to find the right application for a task as well as easy to retrieve, install, and configure the application once an application was found. End users require their applications to be accessible from all client devices they use (e.g. TV, PC, Laptop, Smartphone, etc.), so the applications should not be bound to a certain device or platform. Once a end user purchases a new or additional device the applications should be moved from one device to another easily and quickly [37].

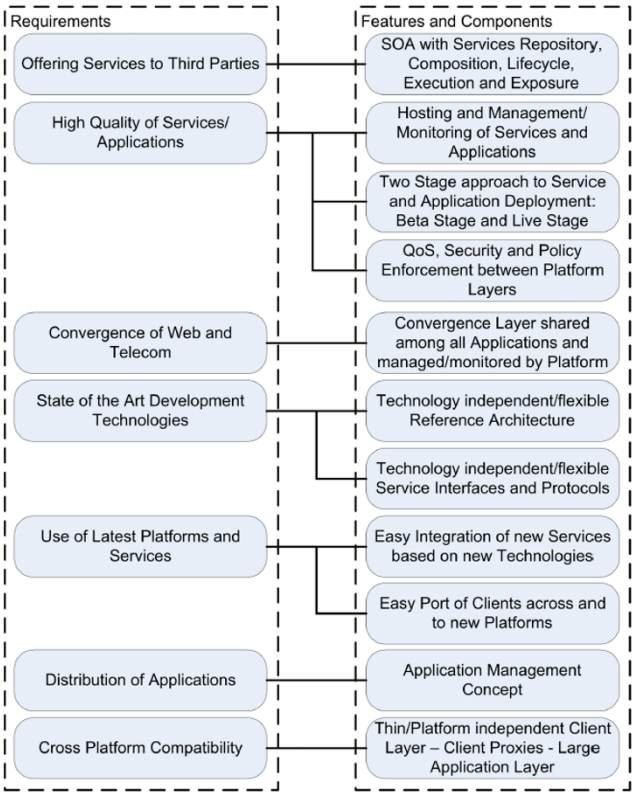

Fig. 2 Mapping requirements to ADP features and components

6 Mapping of stakeholder requirements to required ADP features and components

In order to design an ADP that fulfills the stakeholder requirements, it is necessary to derive the required features and components of the desired ADP from the stakeholder requirements. In Fig. 2, this mapping process is depicted for the central stakeholder requirements and the major ADP features and components that are derived from them.

Offering services to third parties: To offer services to external developers, Web/Internet and Telecommunication services need to be implemented as modular software artifacts and their interfaces need to be exposed to ADP external systems. This can be archived by including a layered SOA concept and by specifying, implementing and deploying all Web/Internet and Telecommunication services as SOA services on the ADP. Further, the SOA concept incorporates additional features such as service composition,

service life cycles, service execution, and service management that are essential for the ADP concept.

High quality of services/applications: To ensure a high quality, it is important to define hosting, management, and monitoring features for services and end user applications on the ADP. This gives the ADP provider all control; he needs to ensure the quality on all levels of the platform for all services and applications deployed on it. Further, it gives the external developers tools to monitor the quality of their software and enhance it steadily.

To differentiate between services and applications with no guaranteed quality and those with high quality that completed extensive testing and monitoring phases, a two stage approach needs to be implemented. (1) Beta Stage services and applications are deployed by developers to share with other developers early, to be tested and enhances further and to be brought to a stable state in the future. (2) Live Stage services and applications are deployed by the ADP provider after an extensive testing and monitoring process and are considered high quality and stable services and applications to be re-used in other applications or to be distributed to end users respectively.

To ensure quality parameters such as security and real-time requirements, the different SOA based layers of the ADP need to be separated by policy enforcement layers with different levels of strictness.

Convergence of Web and Telecom: The convergence must not be implemented on an application layer but on a low and ADP managed/monitored layer that shares the Web-Telecom-Converged Services among all applications on the ADP. The ADP should include tools and functionalities to import Web/Internet and Telecommunication services semi-automatically and to orchestrate the imported services to higher level Web-Telecom-Converged Services.

State of the art development technologies: The choice of implementation and deployment technologies must be made by the ADP vendor that implements and deploys an ADP. Hence, the reference architecture including all features and components must be technology independent and defined as a logical architecture.

The ADP must provide external developers with means to allow the use of their choices of technologies for service interfaces, protocols and application development. Hence, even after defining an implementation and deployment technology for the ADP, the deployed ADP must be technology independent in terms of service interfaces, data exchange protocols, and application client development.

Use of latest platforms and services: The ADP needs to be future proof and needs to allow developers to integrate any new Web/Internet or Telecommunication service based on any new technology easily into the platform on a convergence layer with shared services.

The ADP needs to support the developers at designing their applications in a way so it is easy to port application clients across all client platforms as well as to new client platforms in the future.

Distribution of applications: The ADP needs to provide application developers with means to advertise, distribute, and manage their applications on end user client devices. Hence, existing application store concepts combined with SOA service repository/catalog concepts need to be extended to "meta store" concepts and included in the ADP architecture.

Cross platform compatibility: The ADP needs to provide developers with means to offer application data synchronization and client support across all client platforms to their end users. Hence, it needs to support developers at building as thin and as platform independent clients as possible leveraging client proxies and central application layer business logic parts on the ADP.

7 Holistic SOA based converged applications and application delivery platform (ADP)

The ADP concept needs to apply SOA concepts end-to-end from the Telecommunication and Web/Internet services domain through the Web-Telecom-Converged Applications domain all the way to the end user clients and platforms.

7.1 SOA manifesto principles

To ensure the end-to-end SOA orientation, the architecture concepts are aligned to the SOA Manifesto principles as follows [40]:

Business value over technical strategy: The ADP platform is conceptualized as an open and as much as possible technology independent platform. The platform allows application developers to map business/application goals to services using implementation as well as service interface technologies of their choice.

Strategic goals over project specific benefits: The ADPs overall strategy defines a thorough SOA based approach on platform as well as applications side. This encourages and also forces SOA design and architecture principles for Web-Telecom-Converged Applications hosted and deployed on the platform. In certain projects, this might lead to an overhead in the system architecture design phase but will lead to more re-usable services on the platform and a maintainable and extendable application architecture.

Intrinsic interoperability over custom integration: The ADP concept combines Web/Internet and Telecommunication services on a common platform using standards based technologies that allow immediate interoperability. Applications follow SOA principles, are based on the same technologies as the imported services and so are interoperable by definition.

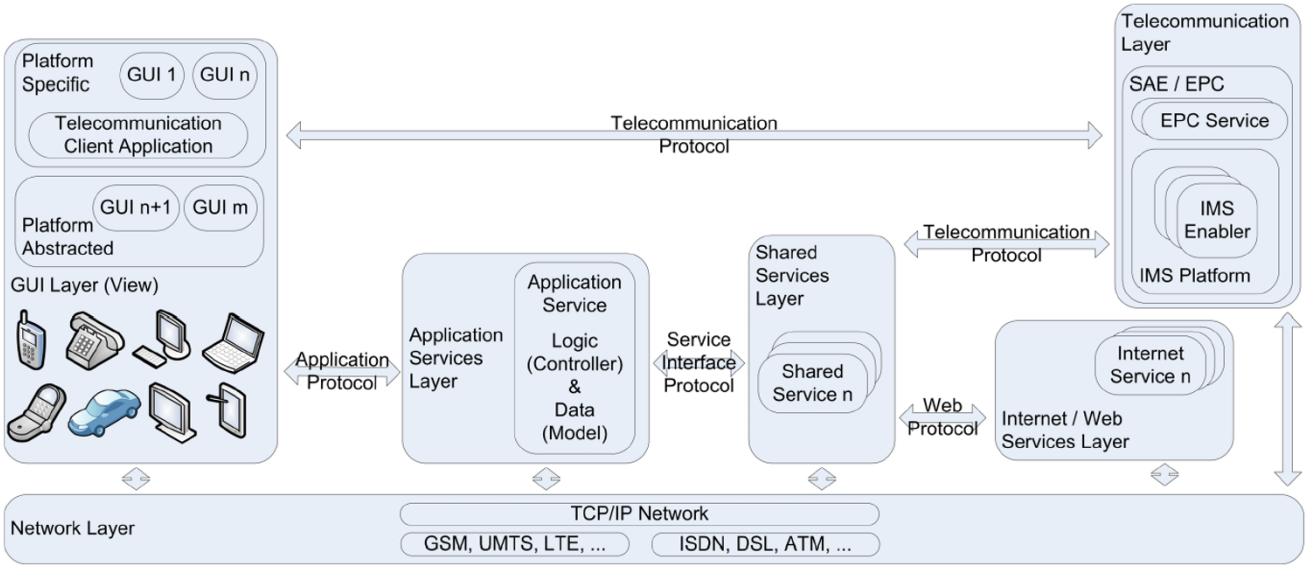

Fig. 3 Logical Web-Telecom-Converged application architecture

Shared services over specific purpose implementations: The ADP encourages and incentivises application developers to identify and encapsulate shared services from their Web-Telecom-Converged Applications during their SOA principles based design and architecture work.

Flexibility over optimization: The ADP platform concept enforces SOA design for all applications and services on the platform and does not allow any optimization shortcuts or deviations from these principles.

Evolutionary refinement over pursuit of initial perfection: The ADP platform defines applications as "application services" and re-usable services as "shared services". Through that, all applications as well as re-usable services can be managed in a service life cycle that allows an evolution/update of all services on the platform over time.

7.2 Distinction between service and application

In this work, a service is defined, adapted from [15–17], as an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities. It is a software artifact designed to support interoperable machine-to-machine (not end user) interaction over a network. It has a clearly defined interface (not an end user interface) described in a machine-processable format and is designed to be re-used by other services or applications. Other systems interact with the service in a manner prescribed by its description using messages conveyed using a defined protocol with serialization in conjunction with other related standards. A service can be deployed and managed by a service life cycle management system such as an SDP.

An application is defined as a software artifact designed to support interoperable human-to-machine interaction through an end user client device. It is not designed to be re-used by any service or other application but it re-uses none to many services. End users have universal access and interact with an application through one to many (graphical) user interfaces for different types of end user clients and platforms.

8 Logical architecture of a generic Web-Telecom-Converged application

The generic Web-Telecom-Converged Application architecture maps the concept of a Model-View-Controler (MVC) [91] architecture to a distributed end user application that uses Web/Internet and Telecommunication services. Figure 3 illustrates the whole architecture.

Telecommunication layer: This layer provides implementations of Telecommunication features. It provides IP based multimedia Telecommunication capabilities such as voice call/streaming, video call/streaming, text/multimedia messaging as well as extended IMS service enablers such as network address book or presence.

Telecommunication protocol: There are several different protocols used to build the connections between the Telecommunication Layer, the Service Layer as well as the native (embedded/platform specific) Telecommunication applications on the GUI Layer. Typical Telecommunication protocols in Next Generation Networks (NGN) and IMS are SIP and Diameter. Typical protocols and specifications used in legacy Telecommunication systems are GSM, UMTS, SS7, etc.

Shared service layer: This layer provides re-usable services that can be accessed using clearly specified service interfaces. These services provide access to Telecommunication functionalities from the telecommunication Layer as well as Web/Internet service functionalities from the Web/Internet Services Layer. The shared services life cycles are managed and monitored by the ADP.

Service interface protocol: These protocols connect the application logic (Controller) of a Web-Telecom-Converged Application with the shared and re-usable services it uses from the Shared Service Layer. Examples used for Web-Telecom-Converged Application are SOAP, REST, HTTP and IP, etc.

Application services layer: This layer hosts the Web-Telecom-Converged Application logic (Controller) and data (Model) as a bundled or packaged software component with optional database technologies. These components represent the central parts of the application and are managed by the ADP. The ADP can be managed by a professional hosting company or a CSP which then ensures that the application meets previously defined high quality standards such as scalability, availability or whatever else the CSP requires from its published customer/end user applications.

Application protocol: These protocols connect the Web-Telecom-Converged Application logic with the GUI Layer, are responsible for data and signaling transport, are application specific and can be any IP based protocol. The ADP supports the application developer at choosing a suitable protocol and provides APIs and interfaces for simple protocol usage of common protocols such as REST, SOAP, JSON, etc.

GUI layer: This layer represents the different GUIs (Views) for the different end user devices the application supports. These GUIs are thin clients for the application logic on the Application Services Layer and can be implemented as native/platform specific (e.g. Java, C++, Objective C, ...) or platform abstracted/independent (e.g. J2ME, Hypertext Markup Language Version 5 (HTML5) [20], BONDI [76], etc.) GUI clients. Independent of the type, all clients are implemented as thin as possible in order to conserve the limited resources on the end device and to be easily ported to other platforms.

Platform specific GUI clients have full access to all client platform APIs and so, to all cutting edge features a device supports. Unfortunately, in order to be ported to other platforms expensive and time-consuming reprogramming of major parts of the clients is necessary.

Platform abstracted GUI clients have reduced access to client platform APIs and so, to a reduced device feature set but are automatically portable to all platforms that support the same platform abstraction technology.

The increasing emergence and importance of platform abstraction technologies and standards (often even solely Web/Browser based technologies) such as HTML5 and BONDI and their steadily expanding access to client platform APIs and functionalities leads to the conclusion that the installation of native applications will soon be obsolete and applications will be delivered in a Software as a Service (SaaS) [92] model.

9 Logical reference architecture of a Web-Telecom-Converged application delivery platform (ADP)

This section presents the logical reference architecture for a ADP. The architecture illustrated in Fig. 4 takes the problems discussed in Sect. 2 as well as a balance of the requirements summarized in Sects. 5 and 6 into account and describes an end-to-end concept for the delivery of Web-Telecom-Converged Application.

9.1 Network layer

This layer is the foundation of all Telecommunication systems and Internet infrastructures and provides all means for service and applications signaling as well as service and application data transport. It includes specifications and standards for mobile Access Network technologies such as GSM, UMTS, Long Term Evolution (LTE) [13] as well as fixed Access Network technologies such as Transmission Controll Protocol (TCP) [30]/IP.

Telecommunication systems based on GSM and UMTS are closed systems with limited/restricted access to and no access from TCP/IP based networks such as the Internet. Advanced Telecommunication system specifications such as LTE are more open and so the interoperability to and from TCP/IP networks is greatly improved. Telecommunication services access is still mostly limited to clients within the network of the respective CSP, but the improved interoperability with TCP/IP removes technical difficulties to access such services from all TCP/IP based networks.

Web/Internet services use protocols such as HTTP that run directly on top of TCP/IP and so provide and deliver their functionality in a universally accessible way for all clients connected to the Internet.

9.2 Telecommunication services layer

Using concepts and specifications such as Next Generation All-IP Networks, Evolved Packet Core (EPC) [14], System Architecture Evolution (SAE) [14], and IP Multimedia Subsystem (IMS) [7], this layer provides an IP based Telecommunication system architecture including control and signaling capabilities for real-time Telecommunication and multimedia services such as voice call/streaming,

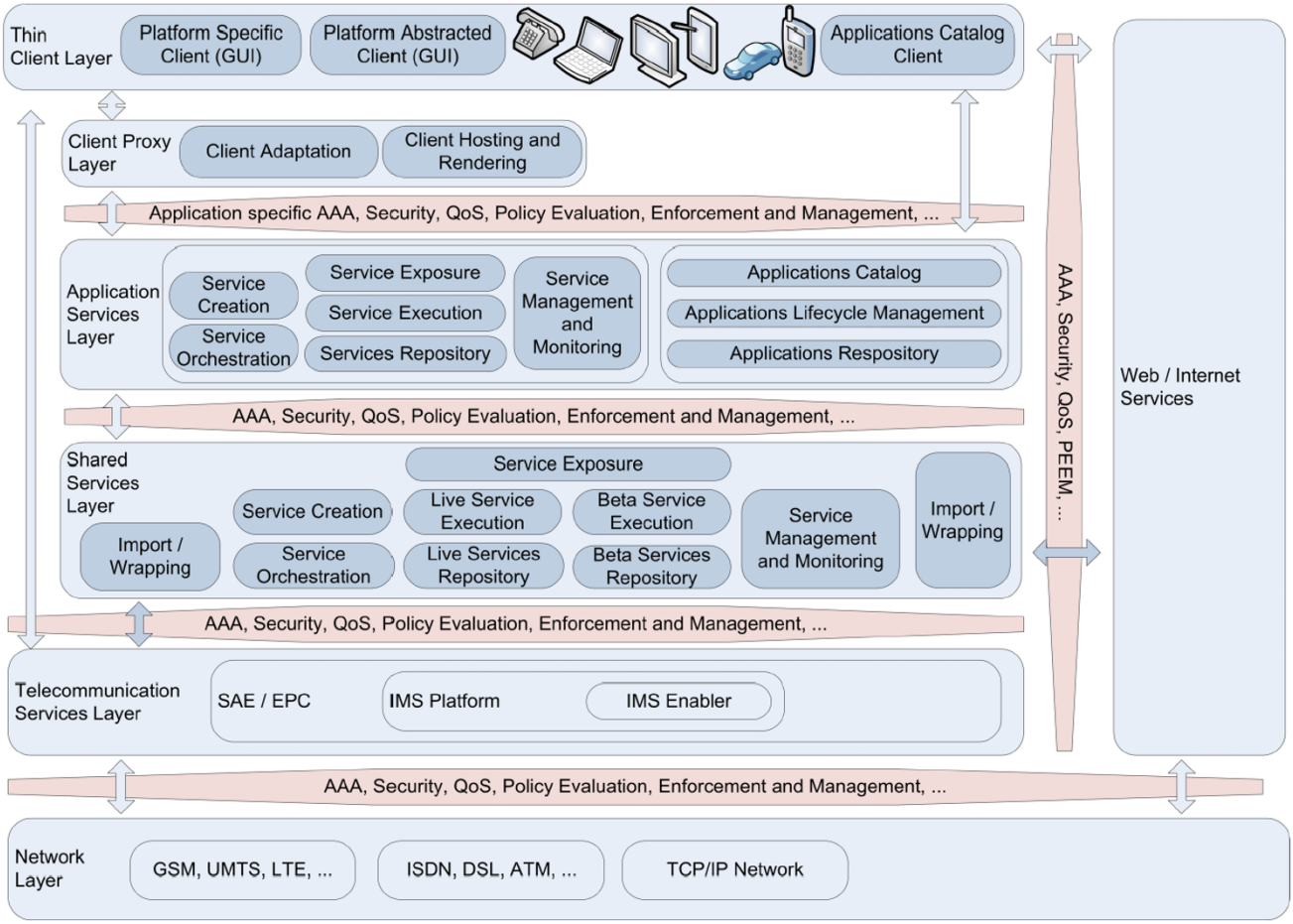

Fig. 4 Logical ADP reference architecture

video call/streaming, text/multimedia messaging, etc. IMS Enablers and IMS application servers provide means for Telecommunication system customization, Telecommunication service extension, and Telecommunication application development.

9.3 Web/Internet services layer

This layer provides access to all TCP/IP based services and applications offered by ASPs on the Internet. Services and applications use OSI Application Layer [28] protocols such as HTTP to exchange data and to provide access for end user clients. Re-usable services use de-facto Internet service interface standards such as REST and data formats such as JSON to expose their functionalities to other services and applications.

9.4 Shared service layer

This layer hosts application independent services that are specified and implemented by developers for developers, ready to be re-used and shared among all applications on the ADP. This layer defines two stages, Beta and Live, for services. Beta Services are still under development and testing or solely experimental. With these services, management and monitoring is limited and a service developer should aim to improve them to the Live stage. Live Services are stable and high quality services that passed a detailed testing and monitoring process. These services are managed and monitored by the ADP to ensure a steady high quality and to be able to react to problems immediately.

On this layer, the convergence of Web/Internet and Telecommunication services takes place. Through import/wrapping and orchestration of shared services that access external Web/Internet or Telecommunication services, Web-Telecom-Converged Shared Services are created.

This layer extends concepts of SDPs as described in Sect. 4.2, in [44, 45, 49] and defines the following components:

Service creation: This component provides features and tools for shared service identification, specification, development (SDKs, APIs, Protocol Stacks, etc.) and deployment on the ADP.

Service orchestration: This component enables developers to compose services from already deployed shared services in a programmatic and descriptive way. This component provides the means to build Web-Telecom-Converged Services by composing basic shared services that access external either Web/Internet or Telecommunication services.

Live/Beta services repository: These components host all shared Live and Beta services on the ADP. They provide access to meta information of the services and their interfaces.

Live/Beta service execution: These components manage the life cycle of all shared Live and Beta services from the Services Repositories on the ADP. They load, start, stop, restart, and archive shared Beta and Live services in conjunction with the Services Repositories.

Service exposure: This component provides Application Services Layer services with access to the shared Beta and Live services through their specified interfaces using their specified interface technologies and data exchange protocols.

Service management and monitoring: This component provides the developers and ADP provider with means to define, manage, and monitor shared Beta and Live service specific quality requirements such as scalability, availability, performance, exceptions, etc.

Import/Wrapping: These components allow the ADP to semi-automatically import Telecommunication as well as Web/Internet services using textual descriptions of their interfaces. The imported services are then, just like any other shared Beta and Live service without access to an external Web/Internet or Telecommunication service, available in the Service Orchestration, Service Execution, and Service Exposure components of this layer. Developers can leverage them to develop re-usable value-added orchestrated Web-Telecom-Converged Services to be used in Web-Telecom-Converged Applications on the Application Services Layer.

9.5 Application services layer

This layer is divided into two parts: (1) It hosts the end user application specific services that implement application business logic and data management and re-use shared services from the Shared Services Layer below. (2) It hosts for every application, version managed application bundles that include the application specific services as well as the distributable end user clients for all platforms the applications supports.

In detail, this layer consists of the following components:

Service creation: This component provides tools for application service development as well as for Shared Services Layer service access (Service Interface Protocol/Technology SDKs and APIs).

Services repository: This component hosts all application specific services of all applications deployed on the ADP. It provides access to meta information of the services and their interfaces.

Service execution: This component manages and monitors the life cycle of application specific services from the Services Repository. It loads, starts, stops, restarts, and archives application specific services in conjunction with the Services Repository.

Service exposure: This component provides clients with access to the application specific services through their specified interfaces using their specified interface technologies and data exchange protocols.

Service management and monitoring: This component provides the developers and ADP provider with means to define, manage, and monitor application service specific quality requirements such as scalability, availability, performance, exceptions, etc. Further, monitoring should be used during application development and, later in the development process, has to be used in conjunction with extensive structured testing to approve an application to be accepted and deployed in the Applications Repository and distributed through the Applications Catalog.

Applications repository: This component stores the packaged application software bundles of approved Web-Telecom-Converged Application containing application business logic and data management as application specific services as well as all end user clients for all supported platforms. In addition to that, the component provides meta data about the applications defined by the developer and/or ADP provider to the Applications Catalog component.

Application life cycle management: This component takes care of application life cycle tasks such as application version management, publication, and distribution of new applications or updated versions, etc. This component interfaces the Applications Repository component for application meta data such as versions, supported client platforms, etc. as well as the Application Catalog component for application distribution, update management, etc. functionalities.

Application catalog: This component gathers and summarizes meta data of all approved Web-Telecom-Converged Applications on the ADP and distributes and updates application clients on the end user client platforms. It provides a browseable and searchable catalog of all applications including enhanced meta information such as descriptions, client platforms support, versions, available updates, user rankings, user comments, price information, etc.

This component enables the Client Layer Application Catalog Client to download and install application clients and updates from the Applications Repository on the end users client platforms.

9.6 Client proxy layer

This layer supports the ADP paradigm of very thin platform dependent or platform independent clients on end user devices. It provides the developers with server side development components that can be used to adapt their platform dependent clients to the specifics of the different client platforms as well as to develop platform independent clients that can be rendered on all client platforms.

It supports the developers at keeping platform dependencies on end user devices as small as possible while still keeping the business logic and data management in the application specific services on the Application Services Layer completely client and client platform independent.

The Client Adaptation component provides adaptation support for platform dependent thin clients. Since client device platforms are very different in terms of screen sizes, processing capabilities, memory, battery power, user interface technologies and concepts, etc., this component allows developers to implement client adaptors on server side that adapt data, interactions, user interfaces, etc. to the specifics of the different client platforms.

The Client Hosting and Rendering component provides a hosting environment for platform independent clients. Such clients are deployed on the server side and only the Graphical User Interface (GUI) is transmitted to and rendered on the client platforms. Examples for that are clients developed using existing Web/Internet technologies such as HTML/AJAX rendered in a browser and upcoming Web/Internet technologies such as HTML5/BONDI, etc. rendered in a browser/widget engine or similar. The component allows developers to deploy adapted versions of their clients or dynamically adapt their clients specifically tailored to different rendering engines on different client platforms.

9.7 Thin client layer

This layer hosts and executes platform dependent and renders platform independent Web-Telecom-Converged Application clients (GUIs) on the end user devices. For application client management, it hosts the client (GUI) of the ADP Application Catalog component that allows end users to search, deploy, and update Web-Telecom-Converged Application clients that are compatible with the device platform.

In addition, some end user devices also run (mostly preinstalled by the device vendor or CSP) native and platform dependent Telecommunication clients such as LTE or UMTS voice/video call clients. These connect directly to the Telecommunication Services Layer in order to access Telecommunication control and signaling system components directly and to implement real-time Telecommunication services access most efficiently (e.g. phone/video call application on a mobile phone).

10 Proof-of-concept technical architecture and prototype

This section presents the two steps that were taken to proof the logical reference architecture of a Web-Telecom-Converged ADP concept in a real software development environment. (1) It presents the results of the mapping of the logical architecture to a technical architecture that was implemented and deployed on a server infrastructure at a research lab. (2) It illustrates the step of this research work, where the implemented, deployed and operational technical ADP architecture is used to implement and deploy a "Converged Address Book" application prototype.

10.1 Technical architecture of the Application Delivery Platform (ADP)

The technical architecture maps all central concepts and components of the logical reference architecture from Sect. 9 to specific technologies, special purpose servers and ADP specific service, application, or tool implementations. Figure 5 provides a visual overview of the results of this mapping.

Telecommunication services layer: This layer represents a future Telecommunication system solely based on IP technology. The central specification to provide multimedia services to end-users is the IP Multimedia Subsystem [7] and in order to use IMS capabilities in this prototype we chose the most widely used open source IMS implementation from Fraunhofer FOKUS, OpenIMSCore [42].

In addition to the multimedia features for end-users IMS specifies additional functionalities such as a Network based Address Book (NAB). This feature is specified as an application of an XML Document Management Server and in order to include this functionality in the ADP prototype, an OpenXDMS [94] is used.

Finally, IMS also specifies a concept that allows developers to extend the core functionalities of an IMS through so called "Enablers". These are defined as SIP applications implemented and deployed on a SIP Application Server (AS). In the ADP prototype this feature is realized through a Apache Sailfin SIP Application Server.

Web/Internet services: In order to include services from the Internet in the ADP prototype, a simple HTTP and REST network gateway connection needs to be open and available. At the moment, for the prototype described below, the Google Contacts service as well as the Facebook Friendslist service are used.

Shared services layer: In order to provide a lightweight and service interface technology independent service life cycle management, OSGi [95] was selected and in form of the JBoss OSGi Container implementation included in the ADP prototype.

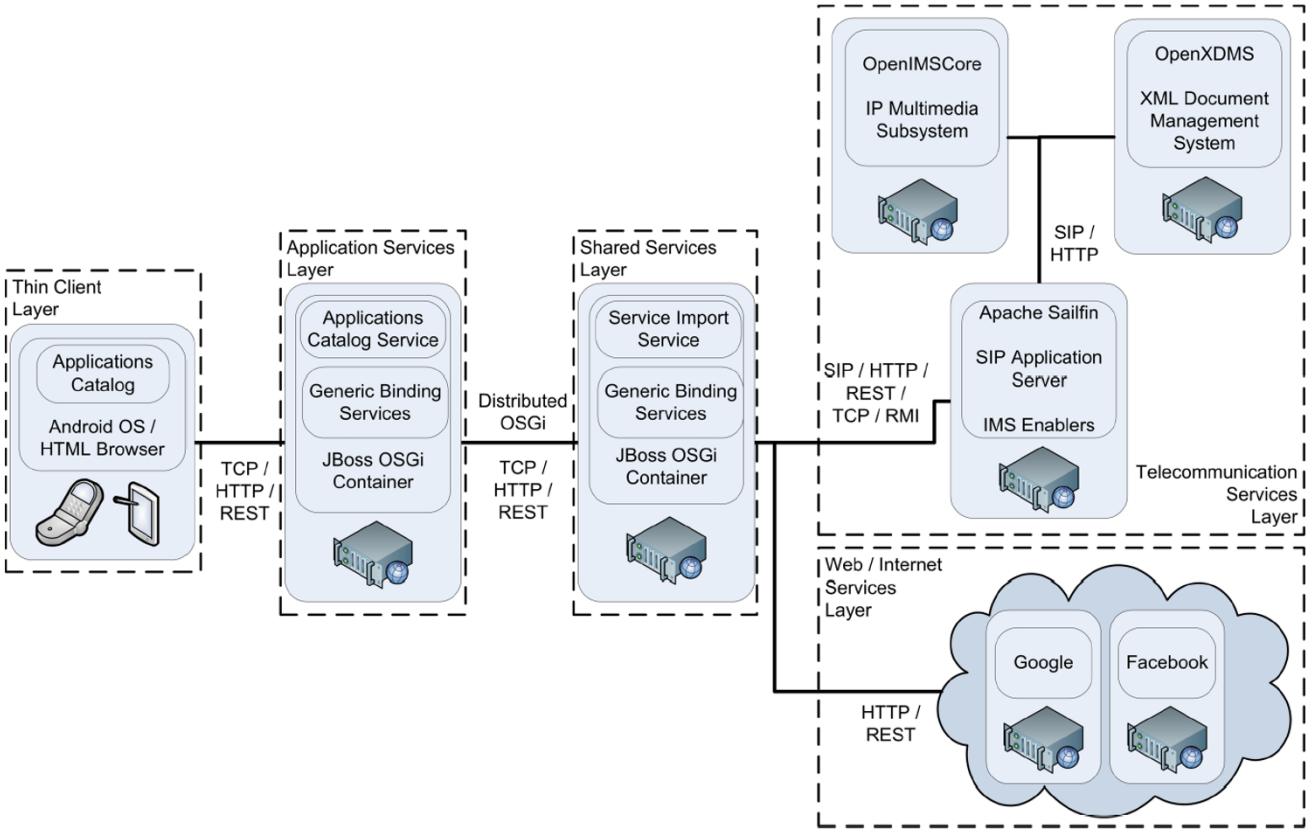

Fig. 5 Technical ADP architecture

This layer takes care of importing/wrapping external services from the Web/Internet and Telecommunication domains. This functionality is provided by the "Service Import Service" deployed as OSGi service on this layer.

Since OSGi is service interface technology independent, the ADP provides several "Generic Binding Service" OSGi services that can be used by shared service developers to bind their service interfaces to specific protocols. In the ADP prototype, only one binding service for REST is implemented.

Application services layer: This layer uses the same technical concept for service management as the Shared Services Layer—JBoss OSGi.

Web-Telecom-Converged End User Applications are deployed on this layer and the "Applications Catalog Service" OSGi service provides an overview over all applications available as well as over meta information, ratings, user feedback and details about client platform compatibility. Further, it manages versioning, installations and updates of application's thin clients on the different end-user client platforms.

Similar to the Shared Services Layer, this layer provides "Generic Binding Service" OSGi services to allow developers to bind service interfaces to protocols of their choice.

Thin client layer: The ADP prototype provides an "Applications Catalog" client for the Android OS and so provides search, installation and updates of Web-Telecom-Converged Applications currently only on Android devices. Nevertheless, applications with a platform abstracted GUI based on HTML or similar are of course accessible from all platforms with compatible browsers.

10.2 Unified address book prototype implementation

For the Web-Telecom-Converged Application prototype implementation, the use case of a Unified Address Book was chosen. The Unified Address Book imports the user's contacts from two Web/Internet sources, Google Contacts and Facebook Friendslist and one Telecommunication source, the IMS Network Address Book of the CSP. The application client is represented by a simple web browser that renders the HTML code returned by an Application Services Layer service that accesses the Unified Contacts Service on the Shared Services Layer. Figure 6 illustrates the service architecture of the prototype implementation.

External services layers: These two layers provide the prototype with access to the contacts data. On the one hand, the Google API and Facebook API are used to retrieve the

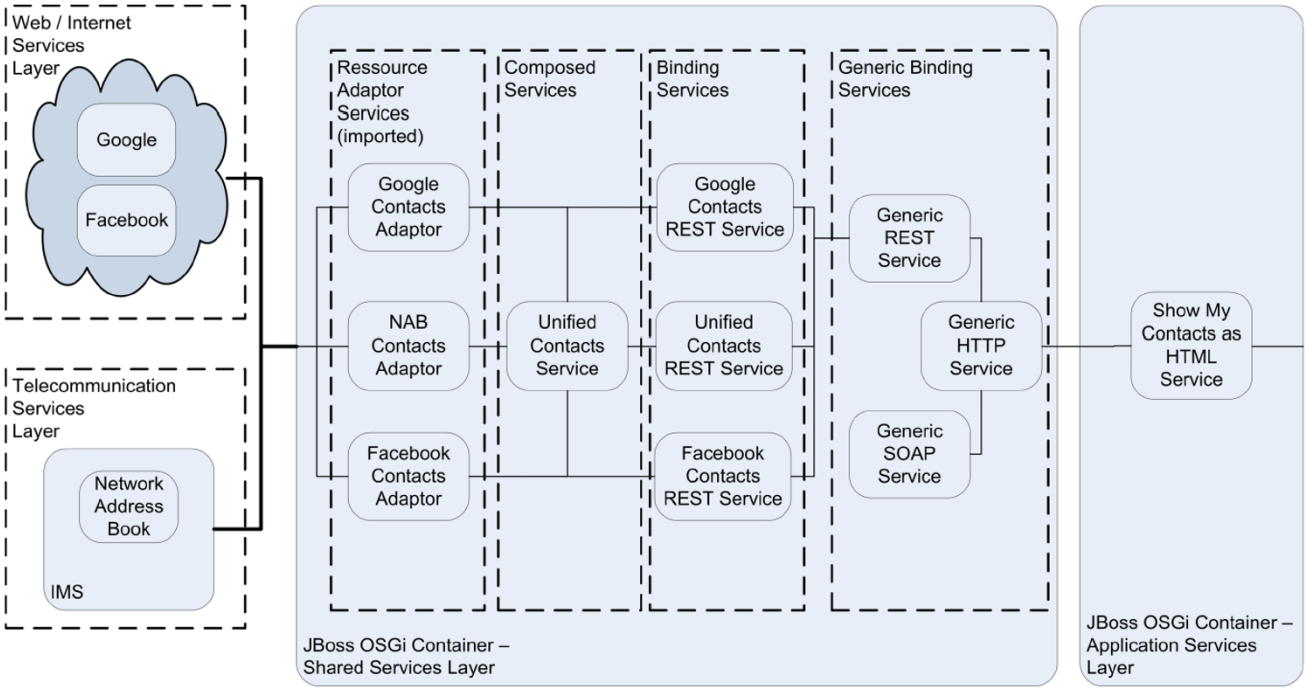

Fig. 6 Converged address book prototype architecture

contacts from these Web/Internet services. On the other hand, the XDMS server based Network Address Book of the CSP's IMS is accessed to retrieve the contacts of the user that are stored centrally within the Telecommunication infrastructure.

Shared services and application services layers: As the figure shows, most services of the application prototype are designed as Shared Services that can be re-used by other applications in the future. All these services are deployed and managed on the Shared Services Layer.

The service that builds the connection to the end user client converts the data from the Shared Services to client specific (in this case web browser) HTML. It is application specific, cannot be re-used by other applications, and so is deployed on the Application Services Layer.

Resource adaptor services: For each external data service, one Resource Adaptor Service is deployed. This service takes care of mapping the external service data to an internal service interface that all services on the Shared Services Layer can look up, reference, and use.

Composed services: In this prototype, declarative service composition such as with BPEL [96], iPOJO [98], SCXML [22, 99], etc. is not yet supported. The service composition is done within the composition service "Unified Contacts Service" on source code level using service container references to the Resource Adaptor Services that are composed.

Generic binding services: Since all services on the Shared Services and also Application Services Layers are interface protocol independent, services that provide a binding to certain protocols are available to be re-used. For this prototype, three different Generic Binding Services are available for the protocols HTTP, REST, and SOAP. Every service on the platform can look them up, reference them and use them in order to bind its interface to a protocol.

Binding services: In order to keep all Shared Services interface protocol independent, protocol binding must not happen on source code level in the Shared Service but within an additional service that couples the interface of the Shared Service with a Generic Binding Service. This is done by the Binding Services. In the prototype, only REST protocol Binding Services are implemented for the Google, Facebook, and Unified Shared Services.

11 Conclusion

This research work identified the current gap between CSP's service delivery and developer platform offerings and developers and end users. It analyzed the goals and requirements from stakeholder viewpoints, combined them with SOA concepts and defined a logical reference architecture for a ADP that closes the gap. Finally, it presented a prototypical technical architecture for a realization of the proposed ADP and a prototype implementation of a Web-Telecom-Converged Application based on the ADP.

The following overview discusses how this research work fulfilled the major requirements identified in Sect. 5.

ADP providers: The ADP is a generic reference architecture that can be implemented, deployed, and provided by CSPs as well as new and independent providers in the Internet or in the Telecommunication industry.

Offering services to third parties: The ADP allows to provide external developers with very high quality services from the Telecommunication as well as Internet domain. The quality of the services and the ease-of-use in combination with existing efforts such as Developer Platforms, that provide tool, business, marketing support, revenue sharing, etc., will lead to a high acceptance at developers.

High quality of services/applications: The ADP provides a ADP provider controlled hosting and management concept for all Web-Telecom-Converged Application on the ADP, and so provides all means for ensuring service and application quality parameters such as scalability, availability, stability on the platform. Further, the ADP provides a two stage approach to service and application deployment where untested services and applications run in a "Beta Stage Mode" and tested, stable, high quality services run in a "Live Stage Mode".

Convergence of Web and Telecom: The ADP enables the convergence of Web/Internet and Telecommunication services on a low, shared and ADP provider managed level. This ensures that not the external developers, but the ADP provider is responsible for the management of these services and ensures and monitors all defined quality parameters.

State of the art development technologies: The ADP reference architecture is completely technology independent and allows an implementation and deployment based on any specific or abstract state of the art technology. The technical architecture and prototypical implementation of the ADP uses service interface and protocol independent Java based OSGI technology that allows the use of the developers choice of state of the art interface and protocol technologies.

Use of latest platforms and services: For developers, the ADP provides all means to include any new and upcoming technology or service into the platform easily, share it and enhance it with other developers, let it be managed by the ADP provider and use it in own applications immediately. Further, due to the ADPs cross client platform compatibility/distribution concepts supporting platform depended and independent thin and thick clients, it allows developers to develop their applications for any new or upcoming end user device with their own choice of level of platform abstraction.

Distribution of applications: Through it's application catalog concepts, the ADP provides developers with CSP access network independent means of distributing and managing applications on end user devices. This also allows end users to easily find the right applications for work or leisure and to manage (install, configure, update, synchronize between devoiced, move between devices, etc.) their applications on their different devices.

Cross platform availability: The ADP architecture concept advocates a Web-Telecom-Converged Application architecture as a distributed system architecture with a large application specific "Application Services Layer" part and as small as possible "Client Layer" part. This fosters the portability of applications through re-use and allows developers to offer their applications on as many end user devices and platforms as possible. This enables synchronization of application data and easy moving of applications between devices by architecture.

As mentioned above, an ADP implementation could be deployed by a CSP, a CSP's subsidiary focusing on innovation and external developer integration or an independent provider. In order to deploy an ADP implementation successfully the provider needs to adopt the mindset of Web/Internet service providers and fulfill all application developer and end user requirements that are out of focus of the ADP.

He needs to allow "quick and dirty" "beta service and application" development and distribution and in order to attract a maximum of application developers, he needs to convey an open and innovative mindset with a maximum of openness, flexibility, support, etc. and a minimum of bureaucracy, binding contracts, costs, etc. The provider needs to convey the advantages of the ADP—a hosted and managed Web-Telecom-Converged Application environment with access to a huge number of ready to use stable services—to developers. He needs to offer incentives and tools for platform contributions, especially on the "Shared Services Layer" and needs to manage all shared services and premium live applications with the highest quality possible.

An ADP deployment will only be successful if application developers fulfill all end users requirements that are out of scope of the ADP and abide by the Web-Telecom-Converged Application architecture recommendations the ADP defines. Further, application developers need to be supported at identifying and distinguish shared from application specific services. They need to adopt an open source community mindset and share services in the whole ADP community among all members and improve shared and application services and applications steadily.

12 Future work

Future work includes the extension of the prototypical realization of the ADP architecture with an extended set of imported Telecommunication and Web/Internet services. Especially real time Telecommunication features such as

voice/video and conference call from the Telecommunication domain as well as instant messaging features from the Web/Internet domain will be included. This will allow the detailed investigation of real time requirements.

In parallel to that, the app store concepts conceptualized by the "Application Catalog" service and client will be implemented for different client platforms such as smart phones, PCs, and IPTVs. Based on that, a "Meta-Store" concept with additional interfaces to existing distribution channels of different platforms (Apple AppStore, Android Market, etc.) will be conceptualized and implemented.

In addition, Service composition in a declarative way by using or even combining technologies such as BPEL [96], SCXML [99], iPOJO [22, 98], etc. including graphical composition tools will be investigated.

Further, the semi-automatic and interface descriptions based import of services from the Telecommunication and Web/Internet domain will be investigated and prototypically implemented on the ADP.

The ADP will be further opened to developers from different projects at our university so they can develop additional prototype applications and Shared Services. Their development experiences, problems, requirements fulfillment satisfaction, etc. will be evaluated. Based on that, a concept for an "Application Development Process using the ADP" will be defined. The ADP prototype will be used to evaluate this development process in future projects.

References

1. GSM Association. Global system for mobile communications (GSM). http://www.gsmworld.com/. Accessed 02.03.2010
2. Institute of Electrical and Electronics Engineers (IEEE). 802.11 wireless local area networks. http://grouper.ieee.org/groups/802/11/. Accessed 02.03.2010
3. International Telecommunication Union (ITU). Very high speed digital subscriber line transceivers (VDSL). http://www.itu.int/rec/T-REC-G.993.1/en. Accessed 02.03.2010
4. The 3rd Generation Partnership Project (3GPP). Voice call continuity (VCC) between circuit switched (CS) and IP multimedia subsystem (IMS). http://www.3gpp.org/ftp/Specs/html-info/23206.htm. Accessed 02.03.2010
5. The 3rd Generation Partnership Project (3GPP). Charging implications of IMS architecture. http://www.3gpp.org/ftp/Specs/html-info/23915.htm. Accessed 02.03.2010
6. The 3rd Generation Partnership Project (3GPP). Quality of service (QoS) concept and architectur. http://www.3gpp.org/ftp/Specs/html-info/23107.htm. Accessed 02.03.2010
7. The 3rd Generation Partnership Project (3GPP) (2006) IP multimedia subsystem (IMS). http://www.3gpp.org/article/ims. Accessed 02.03.2010
8. The 3rd Generation Partnership Project (3GPP) (2011) General packet radio service (GPRS)—enhanced data rates for GSM evolution (EDGE). http://www.3gpp.org/article/gprs-edge. Accessed 02.03.2010
9. The 3rd Generation Partnership Project (3GPP). Universal mobile telecommunications system (UMTS). http://www.3gpp.org/article/umts. Accessed 02.03.2010
10. The 3rd Generation Partnership Project (3GPP). The 3rd generation partnership project (3GPP). http://www.3gpp.org/. Accessed 02.03.2010
11. The 3rd Generation Partnership Project (3GPP) (2005) Location services (LCS) (Release 7). 3GPP TS 22.071 V7.4.0
12. The 3rd Generation Partnership Project (3GPP) (2008) Open service access (OSA) parlay X web services. http://www.3gpp.org/ftp/Specs/html-info/29-series.htm. Accessed 02.03.2010
13. The 3rd Generation Partnership Project (3GPP) (2008) Long term evolution. http://www.3gpp.org/LTE. Accessed 02.03.2010
14. Olsson M et al (2009) SAE and the evolved packet core—driving the mobile broadband revolution. Academic Press, San Diego
15. World Wide Web Consortium (2010) Web of services. http://www.w3.org/standards/webofservices/. Accessed 27.04.2011
16. World Wide Web Consortium (2010) Web services glossar. http://www.w3.org/TR/ws-gloss/#defs. Accessed 27.04.2011
17. World Wide Web Consortium (2010) Web services architecture. http://www.w3.org/TR/ws-arch/#whatis. Accessed 27.04.2011
18. World Wide Web Consortium. Web services description language (WSDL) 1.1. http://www.w3.org/TR/wsdl. Accessed 28.04.2011
19. World Wide Web Consortium. Simple object access protocol. http://www.w3.org/TR/soap/. Accessed 31.01.2011
20. World Wide Web Consortium. Hypertext markup language (HTML) 5. http://dev.w3.org/html5/spec/Overview.html. Accessed 28.04.2011
21. World Wide Web Consortium. Hypertext markup language (HTML). http://www.w3.org/html/wg/. Accessed 28.04.2011
22. World Wide Web Consortium. State chart XML. http://www.w3.org/TR/2010/WD-scxml-20101216/. Accessed 31.01.2011
23. Open Mobile Alliance (OMA). OMA enabler releases. http://www.openmobilealliance.org/Technical/current_releases.aspx. Accessed 02.03.2010
24. Open Mobile Alliance (OMA). OMA service environment (OSE). http://www.openmobilealliance.org/Technical/release_program/ose_v1_0.aspx. Accessed 02.03.2010
25. Open Mobile Alliance (OMA). OMA service provider environment (OSPE). http://www.openmobilealliance.org/Technical/release_program/ospe_v1_0.aspx. Accessed 02.03.2010
26. Open Mobile Alliance (OMA). Client provisioning. http://www.openmobilealliance.org/Technical/release_program/cp_v1_1.aspx. Accessed 02.03.2010
27. Open Mobile Alliance (OMA). OMA browsing. http://www.openmobilealliance.org/Technical/release_program/browsing_v24.aspx. Accessed 02.03.2010
28. Open Systems Interconnection (OSI). Basic reference model. http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip. Accessed 02.03.2010
29. University of Southern California—Information Sciences Institute (September 1981) RFC791—Internet protocol. http://www.faqs.org/rfcs/rfc791.html. Accessed 02.03.2010
30. University of Southern California—Information Sciences Institute (September 1981) RFC793—transmission control protocol. http://tools.ietf.org/html/rfc793. Accessed 02.03.2010
31. Rosenberg J et al (June 2002) RFC3261—SIP: session initiation protocol. http://www.ietf.org/rfc/rfc3261.txt. Accessed 02.03.2010
32. Fielding R (1999) Hypertext transfer protocol—HTTP/1.1. http://www.w3.org/Protocols/rfc2616/rfc2616.html. Accessed 02.03.2010
33. Fielding R (2000) Architectural styles and the design of network-based software architectures. http://www.ics.uci.edu/fielding/pubs/dissertation/top.htm. Accessed 02.03.2010
34. Campbell B et al (December 2002) RFC3428–session initiation protocol (SIP) extension for instant messaging. http://www.ietf.org/rfc/rfc3428.txt. Accessed 02.03.2010

35. Rosenberg J (August 2004) RFC3856—a presence event package for the session initiation protocol (SIP). http://www.ietf.org/rfc/rfc3856.txt. Accessed 02.03.2010
36. Deutsche Telekom. Developer garden. http://www.developergarden.com. Accessed 02.03.2010
37. Deutsche Telekom Laboratories (March 2010) Connected life and work. Internal Presentation
38. Erl T (2005) Service-Oriented Architecture (SOA): concepts, technology and design. Prentice Hall, New York
39. Starke G, Tilkov S (2007) SOA-Expertenwissen—Methoden, Konzepte und Praxis serviceorientierter Architekturen. dpunkt Verlag, Heidelberg
40. Booch G et al. SOA manifesto. http://www.soa-manifesto.org/. Accessed 31.01.2011
41. Deinert F et al (2009) A base solution for exposing IMS telecommunication services to web 2.0 enabled applications. MobilWare
42. Fraunhofer FOKUS. The open source IMS core project. http://www.openimscore.org/. Accessed 31.01.2011
43. Blum N et al (2009) A research infrastructure for SOA-based service delivery frameworks—the open SOA Telco playground at fraunhofer FOKUS. In: 5th international conference on testbeds and research infrastructures for the development of networks and communities, Washington, DC
44. Blum N, Boldea I, Magedanz T, Margaria T (January 2010) Service-oriented access to next generation networks—from service creation to execution. J Mob Netw Appl
45. Blum N, Boldea I, Magedanz T, Staiger U, Stein H (2009) A service broker providing real-time telecommunications services for 3rd party services. In: Proc of 33rd annual IEEE international computer software and applications conference (COMPSAC). Seattle
46. Magedanz T et al (2007) Evolution of SOA concepts in telecommunications. IEEE Comput Mag
47. Alcatel-Lucent. Service delivery environment (SDE). http://www.alcatel-lucent.com/sde/. Accessed 02.03.2010
48. TMForum. Service delivery framework (SDF). http://www.tmforum.org/TechnicalPrograms/ServiceDeliveryFramework. Accessed 02.03.2010
49. Ohnishi H et al (2007) Service delivery platform for Telecom-enterprise-Internet combined services. IEEE GlobeCom
50. Yang J, Park H (2008) A design of open service access gateway for converged web service. In: International conference on advanced communication technology (ICACT)
51. Gbaguidi C et al (1999) A programmable architecture for the provision of hybrid services. IEEE Commun Mag
52. Shao X et al (2008) An integrated Telecom and IT service delivery platform. In: IEEE Asia-pacific services computing conference
53. Kryvinska N et al (2010) A scenario of service-oriented principles adaptation to the Telecom providers service delivery platform. In: Fifth international conference on software engineering advances
54. Agarwal V et al (2005) A service creation environment based on end to end composition of web services. In: International World Wide Web conference, May 2005
55. Hurtado JA et al (2007) A SIP based next generation services platform. In: International conference on mobile ubiquitous computing, systems, services and technologies
56. Burger EW et al (2007) A telecommunications web services platform for third party network access and SOA-based service delivery. In: Workshop on middleware for next-generation converged networks and applications (MNCNA)
57. Jie G et al (2008) A template-based orchestration framework for hybrid services. In: Fourth advanced international conference on telecommunications
58. Lee HJ et al (2010) An Internet-mobile platform for NGN/IMS applications. In: IEEE international conference on E-business engineering
59. Ubiquity Software (2006) Applying SOA principals in developing advanced Telecom applications
60. Baba H et al (2010) Web-IMS convergence architecture and prototype. IEEE GlobeCom
61. Lozano D et al (2008) WIMS 2.0: converging IMS and Web 2.0. designing REST APIs for the exposure of session-based IMS capabilities. In: Second international conference on next generation mobile applications, services, and technologies
62. Bo C et al (2010) Design and implementation of web-Telecom hybrid services bus based execution platform over convergence networks. IEEE GlobeCom
63. Bo C et al (2010) Development of Web-Telecom based hybrid services orchestration and execution middleware over convergence networks. J Netw Comput Appl
64. Zhou YC et al (2010) Storm Service: A self-service telecommunication service delivery platform with platform-as-a-service technology. In: IEEE sixth world congress on services
65. Farley P, Capp M (2005) Mobile web services. BT Technol J 23(2)
66. Pollet T et al (2006) Telecom services delivery in a SOA. In: 20th international conference on advanced information networking and applications (AINA)
67. Baravaglio A et al (2005) Web service applicability in telecommunication service platforms. In: International conference on next generation web services practices (NWeSP)
68. T. O'Reilly (Sept 2005) What is web 2.0. http://oreilly.com/web2/archive/what-is-web-20.html. Accessed 02.03.2010
69. The Symbian Foundation. Symbian OS. http://www.symbian.org/. Accessed 02.03.2010
70. Apple. iPhone OS. http://developer.apple.com/iphone/. Accessed 02.03.2010
71. The Linux Foundation. Linux. http://www.linuxfoundation.org/. Accessed 02.03.2010
72. International Telecommunication Union (ITU) (Oct 2006) Proposed definition and description of IPTV services for IPTV service scenario
73. Google. Android. http://developer.android.com/. Accessed 02.03.2010
74. The Linux Foundation. MeeGo OS. http://meego.com/. Accessed 02.03.2010
75. Microsoft. Windows Phone 7. http://www.windowsphone7series.com/. Accessed 02.03.2010
76. BONDI. BONDI—open mobile terminal platform. http://bondidev.omtp.org/default.aspx. Accessed 28.04.2011
77. PJSIP. Open source SIP stack and media stack for presence, im/instant messaging, and multimedia communication. http://www.pjsip.org/. Accessed 02.03.2010
78. MJSIP. A complete java-based implementation of a SIP stack. http://mjsip.org/. Accessed 02.03.2010
79. Crockford D (2006) The application/JSON media type for JavaScript object notation (JSON). http://www.ietf.org/rfc/rfc4627.txt. Accessed 02.03.2010
80. XML-RPC (2003) Extensible markup language remote procedure call. http://www.xmlrpc.com/. Accessed 02.03.2010
81. GigaOM (Jan 2010) The apple app store economy. http://gigaom.com/2010/01/12/the-apple-app-store-economy/. Accessed 02.03.2010
82. Detecon (2010) Future Telco service architecture market analysis. Deutsche Telekom Laboratories
83. Baset S, Schulzrinne H (2006) An analysis of the skype peer-to-peer Internet telephony protocol. In: INFOCOM
84. QuEST Forum (1998) The Telecom quality management system. http://www.tl9000.org/. Accessed 02.03.2010
85. Vodafone RD Lab. Beta Vine. http://www.betavine.net. Accessed 02.03.2010
86. France Telecom SA Orange. Orange Partner. http://www.orangepartner.com. Accessed 02.03.2010

87. Mottishaw P (September 2009) Next generation service delivery platforms. http://www.hp.com/go/sdp. Accessed 02.03.2010
88. Google. Android market. http://www.android.com/market/. Accessed 02.03.2010
89. Apple. App Store. http://www.apple.com/iphone/apps-for-iphone/. Accessed 02.03.2010
90. Garrett JJ. Ajax: a new approach to web applications. http://www.adaptivepath.com/ideas/e000385. Accessed 31.01.2011
91. Singh I et al (2003) Designing enterprise applications with the J2EE platform. Sun Microsystems Inc
92. Software and Information Industry Association (2001) Software as a service—strategic backgrounder. Software and Information Industry Association
93. GSMA. OneAPI. http://www.gsmworld.com/oneapi/reference_documentation-version_1.html. Accessed 31.02.2011
94. JBoss Mobicents. Mobicents XML document management server. http://hudson.jboss.org/hudson/view/Mobicents/job/MobicentsBooks/lastSuccessfulBuild/artifact/sip-presence/index.html#xdms-XML_Document_Management_Server. Accessed 31.01.2011
95. OSGi Alliance. The OSGi service platform. http://www.osgi.org/Main/HomePage. Accessed 31.01.2011
96. OASIS. Web service business process execution language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. Accesses 31.01.2011
97. Chappell D (June 2004) Enterprise service bus. O'Reilly
98. Apache Felix. iPOJO. http://felix.apache.org/site/apache-felix-ipojo.html. Accessed 31.01.2011
99. Moro S et al (2008) Service composition with real time services. ICIN