Journal of Mobile Multimedia, Vol. 6, No.3 (2010) 185-206

A PLATFORM FOR USER GENERATED MULTIMEDIA COMMUNICATION SERVICES

NIKLAS BLUM THOMAS MAGEDANZ

Fraunhofer Institute FOKUS
{niklas.blum, thomas.magedanz}@fokus.fraunhofer.de

HORST STEIN INGO WOLF

Deutsche Telekom Laboratories
{horst.stein, i.wolf }@telekom.de

Received December 16, 2009
Revised May 4, 2010

Existing telecommunications networks and classical roles of operators are subject to fundamental change. Many network operators are currently seeking for new sources to generate revenue by exposing network capabilities to third party service providers. At the same time we can observe that applications in the World Wide Web (WWW) are becoming more mature in terms of communications functionality and the definition of APIs that are offered towards other services. The combinations of those services are commonly referred to as Web 2.0 mash-ups. Rapid service design and creation becomes therefore important to meet the requirements in a changing technology and competitive market environment. This paper describes our approach to enable a service creation environment for complex, orchestrated real-time communications services through a service broker on top of Next Generation Networks (NGN) to combine the emerging web/telecommunications service space with existing network provider infrastructures.

Key words: Service Creation Environment, Service Delivery Platform, Service Modelling, Next Generation Network, Service Oriented Architecture, Mash-up.
Communicated by: D. Taniar

1 Introduction

The programmable network envisioned in the late 1990s within standardization and research for the Intelligent Network is coming into reality using IP-based Next Generation Networks (NGN) and Application Programming Interfaces (API) for open developer access. Furthermore, the emerging Web 2.0 digital marketplace presents an important opportunity for telecommunications operators to sell their own capabilities as services. Besides communication services like Short Message Service (SMS), voice-call, video conference, further capabilities like billing, identity management, authentication or control of Quality of Service (QoS) are candidates as telecommunications enablers for the development of new applications.

Deutsche Telekom AG has launched its open development portal developer garden [1] in November 2008 starting with communication related enablers like SMS, voice call, IP location and

conferencing. The target of the portal is to expose a simple Application Programming Interface (API) for developers, who use these services as enablers to create new integrated services. Via the developer garden, the third party service developer gets access to core network functionalities using Web Services and the open programming interfaces via a Software Development Kit (SDK). Comparable activities are performed in British Telecom's Ribbit [2] and in the Orange Partner programme [3].

Moreover, subject matter experts working in several business environments or even end users are increasingly bringing up pressure to telecommunications and Web Services providers by requiring innovative and attractive services, which need to be integrated into a new service life-cycle for development, integration, testing and deployment at a faster pace. Operators reflect this fact that consumers and end users represent an important group of "partners" in developing new applications [4]. We believe on the one hand that operators exposing telecommunications services will need to offer APIs tailored to fit the developer's needs to succeed in a competing open service provider market. On the other hand, for technical non-experts it is required to offer an environment to create and orchestrate heterogeneous telecommunications and non-telecommunications service end points by integrating 3rd party APIs together with communication specific service end points.

Applying this concept to the topic of service development, we present a mash-up development studio attached to a service delivery environment. Above shown Figure 1 illustrates the workflow we have implemented.

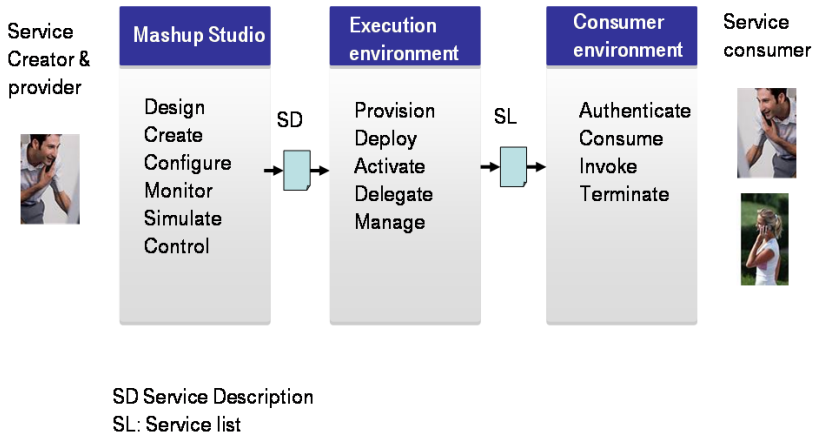

Figure 1 User-generated Service Workflow.

The bases of mash-ups are configurable service elements using telecommunications enablers or other IT Web Services. The combination and configuration of the service elements in form of a visual data flow oriented mash-up style is explored in other contexts (e.g. Yahoo! Pipes [5], Lego Mindstorms [6], Scratch [7]). The end user - having no special programming skills - is enabled to develop, assemble and activate his services and share them with his partners or friends. We support the exposure of the generated services in multiple environments, e.g. social networks via opensocial widgets, web portals, or mobile widgets. Figure 2 depicts a classification of enablers and shows some examples we realized as basic services in our execution environment.

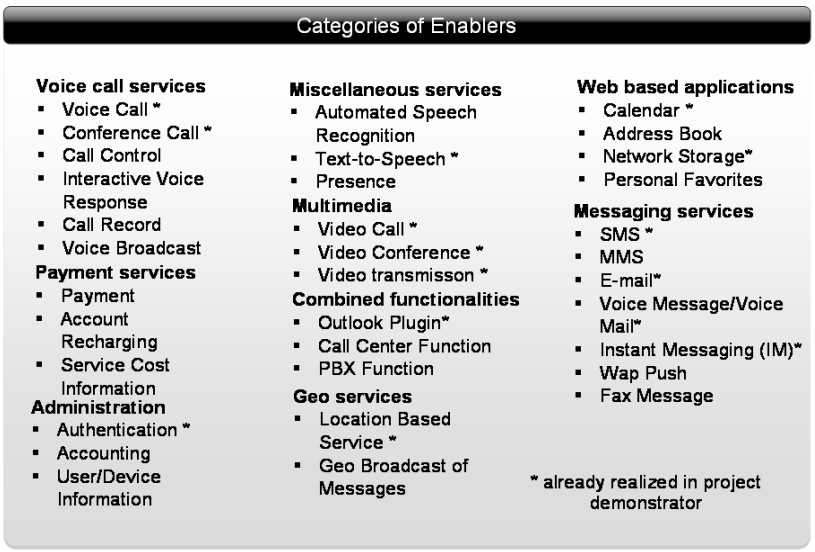

Figure 2 Realized basic services mapped to enabler categories.

For the telecommunications operator it is highly relevant to manage these exposed service enablers i.e. valuable sources of functionality and content to a wider audience either used from developer or from end users. Therefore the operator has to underpin the service delivery platform with certain security and management mechanisms. A set of technologies must be provided in order to effectively abstract from specific network technology, control the access to the service, manage the impact of the service usage, apportion this payment according to revenue sharing rules if the service is partly or wholly owned by a third party provider.

This paper describes research results of our mash-up studio, execution environment, and service broker allowing the definition of request- and service-specific policies serving as Service Level Agreements (SLAs) between a service access gateway to applications enablers in an operator environment and cloud-based applications in third party domains at Deutsche Telekom Laboratories in cooperation with the Fraunhofer Institute for Open Communication Systems (FOKUS).

The paper is structured as follows: section 2 provides a brief overview of related standards and technology, section 3 and 4 focus on our design objectives and on-going related work. Section 5 describes the architecture of our system, focusing on the mash-up studio for service creation, the service execution environment, and the service broker. Section 6 depicts example applications to validate our system. We conclude our work in section 7 and provide an outlook for future work.

2 Related Standards and Technologies

In this section the applied network abstraction concepts for telecommunications are introduced, namely Parlay X, Next Generation Network (NGN) service enablers, and the Open Mobile Alliance (OMA)

Service Environment. Furthermore the related standards out of the SOA world that are relevant to the project architecture are briefly described

2.1 Network Abstraction and Telecommunications Service Enabler

The Intelligent Network was driven conceptually by exploiting the Remote Procedure Call (RPC) and functional programming, whereas the IT world has moved towards object oriented programming languages like C++ and Java that have led to new middleware concepts allowing the implementation of scalable and distributed service delivery platforms and providing abstraction from the details of underlying network signaling and transport protocols.

Application Programming Interfaces (APIs) for telecommunications have been introduced in the mid 90's and basically had the same targets as the Intelligent Network (IN) but are based on newer IT concepts [8]. The related telecommunications API standards, such as Parlay and the 3GPP Open Service Architecture (OSA) [9] and the Java APIs for Integrated Networks (JAIN) [10] aimed for making telecommunications service implementation easier compared to the traditional IN by abstracting from the signaling protocol details of the underlying telecommunications networks (such as the ISDN User Protocol (ISUP), SIP [11], or even the IN Application Protocol (INAP), etc.). These APIs have been defined in a market oriented way over the last years and feature telecommunications related capabilities, such as call control, conferencing, messaging, location, charging, presence, group definitions. The latest standards of OSA/Parlay are the so called Parlay X APIs that provide simplified SOAP-based access to telecommunications-specific services.

The basic idea of these APIs was that Application Servers (AS) host the application logic and access via a secure network connection these APIs which are provided by a dedicated network service gateway at the operators domain. The Application Servers could be operated by the network operator itself or eventually depending on the business model by third parties. In regard of the later approach, network capabilities could be exposed to third parties to take advantage of their creativity and create win-win business models for both operators and application providers in case the APIs are globally standardized and published and competition will be established among operators and application providers.

Similar to service independent building blocks which form part of the conceptual model for Intelligent Networks, the Open Mobile Alliance (OMA) has defined service enablers for the NGN-based IP Multimedia Subsystem (IMS) [12]. The idea was initially born during the specification of a Push-to-Talk over Cellular (PoC) [13] service, a walkie-talkie like communication service between several mobile peers. PoC uses Presence, Group Management and Instant Messaging as enablers to provide information to the users as well as to the PoC service. This led alongside the standardization of PoC to the definition of Presence SIMPLE [14] for Presence and Instant Messaging and XML Documents Management Server (XDMS) [15] for group and list management.

These service enablers provide, abstracted from specific protocols and underlying networks, basic telecommunications service capabilities that can be re-used by other applications and accessed either via protocols or APIs.

2.2 OMA Service Environment

The definitions of several application service enablers by the OMA and the need for a general access function for third party service access led to the specification of the OMA Service Environment (OSE) [16] as a common abstraction layer for IMS-based NGNs. It defines access to an enabler layer which incorporates specific enabler components that offer interfaces to services that implement certain application logic. OMA does at the point of writing not standardize any mapping to a specific middleware messaging technology but leaves this open to the implementation of specific service environments.

The Policy Enforcer or Policy Evaluation, Enforcement and Management (PEEM) component as the function has been named officially by the OMA can be used to intercept service requests from a foreign domain as well as from any other service requestors and apply certain rules (policies). In the first place policies may be used for the authorizations of requests. PEEM may furthermore define enabler capabilities for exposure based on request policies. Depending on the business model different charging rules may also be applied through specific policies. In this regard the definition of policies may be considered as an expression of service contract between a network operator and a service provider.

The OMA names two different Policy Expression languages, Common Policy by the Internet Engineering Task Force (IETF) [17] for authorization policies and Business Process Execution Language (for Web Services) WSBPEL 2.0 defined by Advancing Open Standards for the Information Society (OASIS) [18] for the orchestration of enablers. As by standardization of the OMA, a PEEM provides functionality for service authorization and delegation of service requests depending on policy evaluation.

2.3 Service Orchestration

One can observe the general trend in industry towards the realization of service oriented architecture principles in the last years. Big industry players [19] as well as the open source community [20] recently came up with implementations of the most important and widely accepted SOA standards using Web Services:

- Web Services Definition Language (WSDL 1.1) [21]
- Simple Object Access Protocol (SOAP1.1) [22]
- Web Services Business Process Execution Language [18]

Another candidate is missing here on purpose: Business Process Modeling Notation from the OMG (BPMN) [23] It provides a graphical representation suitable for visual programming of complex business processes, however it is targeted to programmers as opposed our projects design objectives.

3 Design Objectives

The design objectives for our service environment are:

- To provide a high-level service creation environment to non-experts for communications services.
- To be transparent to services it controls. For example, it enables third party service provider to select, invoke and compose services and underlying service enabler can be added, deleted, and moved by the operator without knowledge of the service request originator.

- To provide a flexible environment for service contract and Service Level Agreement (SLA) definition
- To alter service requests and responses based on constraints, e.g. allow dial-out to certain country codes or to define certain accuracy for location information.
- To ease telecommunications service mash-up, e.g. to provide one single anchor point for service creation and accessibility of services.

To provide a service capability set to service developers and third party services, e.g. to provide an interface for service creation environments (SCEs) to dynamically query available services in the operator network.

4 Related Work

Service orchestration for NGN and Web Services has been studied extensively in recent years ([24] and [25]). Authors in [26] discuss blending HTTP, RTSP, and IMS. Authors in [27] discuss the interoperability between IMS and Web Services within the IMS domain for next-generation mobile networks and authors in [28] propose to use SIP-based micro service orchestration and web service bus to seamlessly integrate IMS, Web Services and the underlying services.

Service creation plays a major role in Internet Telephony because it enables openness and programmability by offering frameworks for the development. Authors in [29] provide an overview of service creation technologies for IP-based telephony that still require knowledge that non-experts may not have and present a graphical service creation environment for JAIN SLEE environments. Authors in [30] present a case study of Parlay X service composition using a model-driven approach and authors in [31] with a similar approach using a web-based service creation toolkit to address especially small and medium enterprises (SMEs).

Model-driven composition of networks and services based on SOA principles are in scope of research as part of the European Future Internet Research & Experimentation (FIRE) [32]. As discussed at the Future of the Internet Conference the authors of [33] proposed a platform for graphical service creation targeted at individuals with no specific skills in computer science or programming and aiming at a service-oriented execution environment capable of a seamless interoperation of Web Services and telecommunications applications based on operator-owned infrastructure.

The existing solutions for orchestration are mainly focusing on addressing specific applications, rather than generic solutions. It is difficult to compare the advantages and disadvantages of various solutions, because they are usually available to certain scenarios or specific domains only. The mentioned activities related to service creation are limited in terms of proven correctness of the generated executable software.

Compared with the above-mentioned schemes, our objective is to propose a generic controllable but transparent solution of exposing and developing real-time communications services. To the best of our knowledge an entity that allows the definition of different roles for operators, service providers and developers providing specific service capabilities and service level agreements by a more generic perspective on a developer / service provider orientation with a single mechanism remains untouched.

5 General Architecture

The service environment consists of three main layers as depicted in figure 3:

- Service Creation Environment (Mash-up Studio)
- Service Execution Environment / Service Broker
- Service Delivery Network (incl. network abstraction)

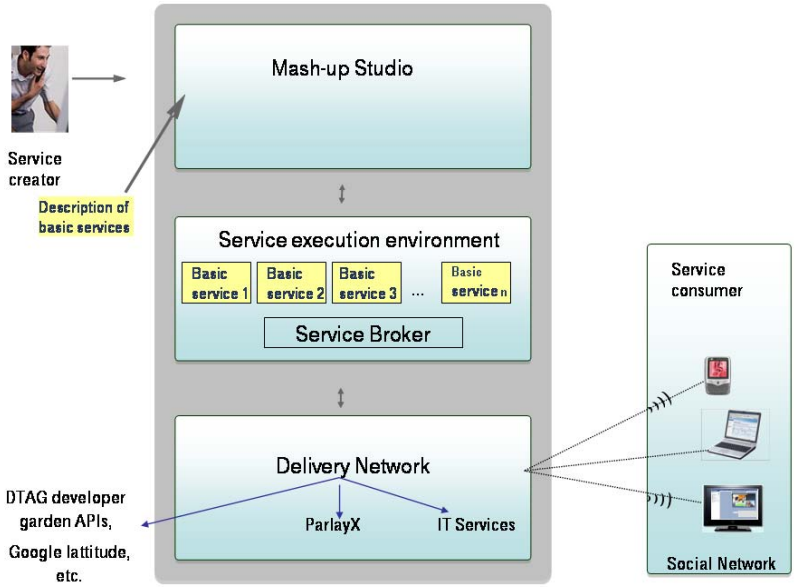

Figure 3 Three Tier Architecture.

The user composes the application in the mash-up studio called Service Creation Workbench (SCW). The service execution is controlled and performed in the Service Execution Environment (SEE) as second layer. Basic services (see figure 2) are realised as Web Services in this layer which map to enabling resources in the delivery network. The third layer provides the access to network resources. The network control and transport to different devices is conducted via a network abstraction and related service enabler. Each of the three tiers is described in this section in more detail.

5.1 Mash-up Studio

The significant simplification of the service creation process is an important step for software engineering methodologies. If the effort for service development can be reduced, the developer can focus more on the problem itself instead of dealing most of the time with the requirements and the details of the technical background. Furthermore and most important for our approach, non-experts are able to create new services themselves.

Therefore the procedure model takes into account that services will not only be designed by software experts, but also by non-experts in the Information and Communications Technology (ICT)

domain. The service creation environment provides all functions for the non-experts to create and deploy and their domain specific services. The main working tool to achieve this is the Mash-up Studio. The following figure 4 provides a screenshot of the Mash-up Studio:

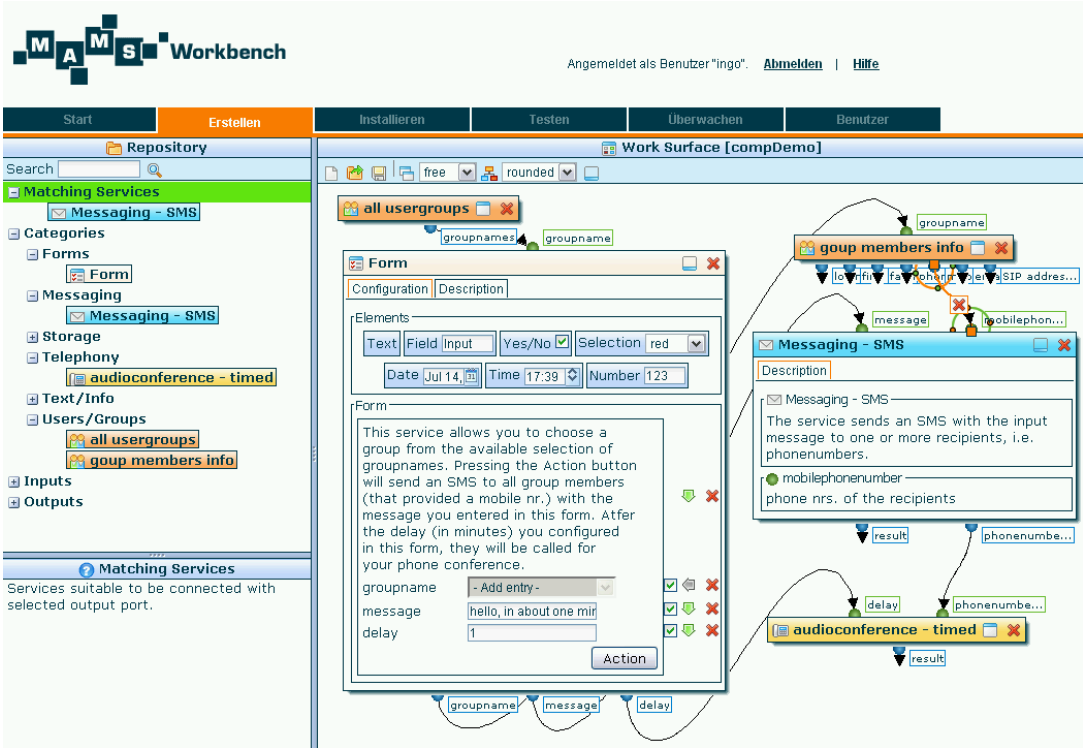

Figure 4 Screenshot of Mash-up Studio.

The Mash-up Studio consists of a graphical user interface which allows the service creator to develop a data flow oriented description of different preconfigured services. The graphical description of the services with boxes, links and parameters is mapped to a Service Description Language (SDL) ([34] and [35]) which also comprises control constructs (e.g. filter, logical operations). Figure 4 provides a screenshot of the web-based Mash-up studio with an example service flow for an aggregated service which allows an end user to initiate a phone conference with selected persons and announce the conference with an invitation via SMS. The service composition consists of basic services for conference call, SMS, a web dialog for selection of conference participants from a user list.

The pre-configured services are the building blocks within our service creation environment. They are represented as items of a service repository (left side in figure 4) which can be fetched and placed via drag and drop on a graphical work area. Once an item is placed on the work area it expands to a box with input and output ports. Connections between the boxes are simply built by drawing links between the corresponding output and input ports of two boxes. We developed an interface system with simple types (string, integer, date, etc.) for input and output ports, which supports the user during the selection of connectable services.

The basic services themselves are described in a self-defined eXtensible Markup Language (XML) format as instances of our intermediate model. A major requirement for this intermediate model was to incorporate standards to the highest possible degree while being independent of any specific service execution environment. Therefore this model was defined using XML Schema for the purpose of aggregation of all necessary data that make up a service composition. It acts as a container during the service creation process and holds Web Service Description Language (WSDL)-based service interface definitions as well as XForms instances for graphical representation of each basic service.

The service creator configures the basic services (e.g. define the target user group, define preferred communication media, etc.) and combines the basic services to aggregated services, which fulfil the creators / service provider's service requirements. Basic services are connected via data links which are evaluated during service deployment. Out of the intermediate description of the composition that is fully based on standard SOA technologies (WSDL and BPEL) a specific deployment package can be generated, deployed and executed on a runtime environment that is compliant with those standards.

A big advantage of sticking with the SOA standard technologies and having the composed service described independently of any specific service execution environment becomes obvious at this point. The generation of a deployment package becomes a substitutable software module that addresses the specificities of a certain service execution environment. Additional functions to monitor, cancel or delete services are provided to the service creator.

The aggregated services can be exposed to different channels (web portals, social networks, widgets, etc.) and are realised with an interface to social networks which support the opensocial interface definition.

5.2 Service Execution Environment

The Service Execution Environment (SEE) manages the operation of the composed service within the framework and can be seen as core runtime environment for the service logic. Based on the principles of service oriented architecture, the SEE integrates several different communication infrastructures and features from different service domains.

In this context, features are also service components from a different domain that might be triggered from telecommunications perspective by incoming events as calls or messages that trigger a service enabler from another service provider's domain, i.e. a Web 2.0 service enabler as maps and/or information sources.

The consequence is that features need to be integrated into service creation environments (SCEs) that are not part of the service provider's domain but available via remote procedure calls on the WWW. This also means that during run-time of complex services involving non-operator enabler, application misbehavior might occur through feature triggering by non-communication services.

To reduce such risks for the operators, we have defined our policy-based service broker according to OMA PEEM specifications that serves also as anchor point for SCEs.

The Service Broker is designed as several independent modules that are accessible through OMA compliant interfaces, especially PEM-1 [36] and PEM-2 [37] interfaces to trigger policy evaluation requests for intercepted service requests and for policy retrieval from the policy rep*osito*ry [38]. Figure

5 depicts our approach of integration of Parlay X and into the SCE and the role of the service broker for the SCE as well as during execution time:

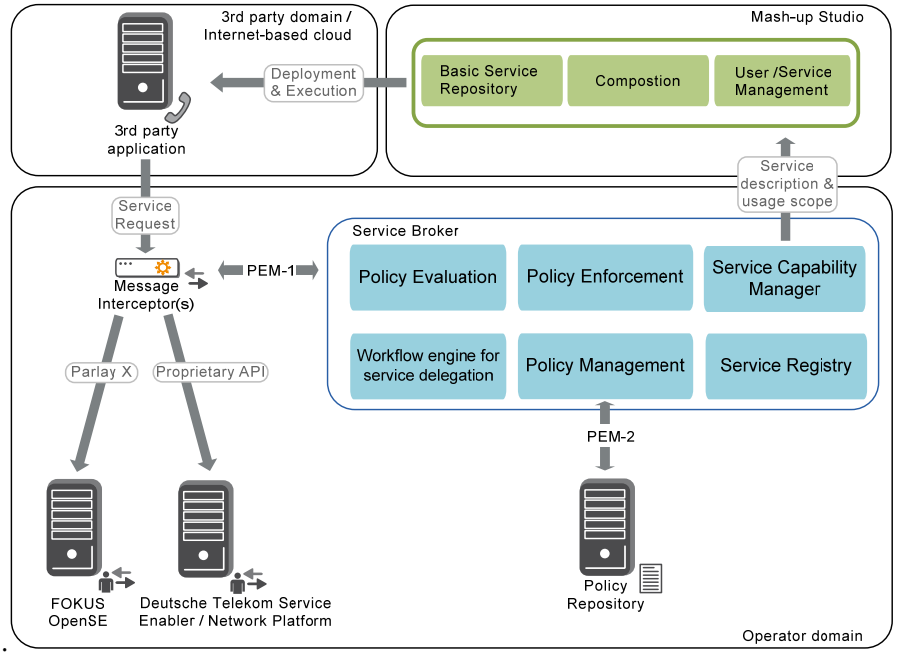

Figure 5 Service Enabler, Service Broker and Service Creation Environment.

Message Interceptor(s) may be applied on all critical interfaces (e.g. Web Services) within an operator's architecture and acts as an intermediate system between the resource requestor and target resource. Upon intercepted messages, messages will be transformed to PEM-1 compliant templates to be transmitted to the service broker. Based on the evaluation results (allow, deny, forward), this subsystem decides either to forward the message to a destination or reply to the originator with a deny response. We decided for this distributed Interceptor/Broker functionality as this approach allows us to keep the broker generic in regard of protocols and specific services and allows us to extend the scope towards SIP and e.g. HTTP later.

The Callable Interface represents the PEM-1 compliant interface exposed by the Service Broker to the Message Interceptor or to other resources that might issue evaluation requests directly. The information received over this interface represents the input for further Policy Evaluation and Policy Enforcement Engine processing.

Policy Evaluation Engine represents the main component of the service broker as it identifies the associated policies of the processed message, evaluates their rules conditions and initiates the execution of the related actions, if needed. The Policy Evaluation Engine identifies the policies based on a predefined application identifier and a combination of the policy identifiers provided by the processed message.

The evaluation result of conditions translates into a possible decisions of allow, deny or forward. In case one rule evaluates to true, the evaluation process will continue with the evaluation of the other associated rules and furthermore with the rules' actions execution. Otherwise, the message is denied and the evaluation of further rules' conditions is abandoned.

Relevant actions set execution consists mostly in delegation of processing responsibility to other resources and transformations of the processed message requests or its response. As the service broker was designed for access control, each delegation action will be evaluated before being sent to destination. This operation mode also ensures that the SLA (specified as a policy) defined between the (mobile) operator and the third party will be respected.

The Policy Enforcement Engine takes over the responsibility of actions execution and authorization decision from the Policy Evaluation. Furthermore the Policy Enforcement Engine sends the authorization decision to the Message Interceptor through Callable Interface reply messages based on the Policy Evaluation Engine processing result and the data returned by the associated actions execution.

From our perspective, a policy is a formalism used to describe how to manage resources, to provide a controllable access and to reuse the existing resources exposed by an operator or third party provider. Each policy is defined as a rule set and each rule is composed of conditions and actions. Actions may be either identifier for responses of the Policy Enforcement Engine to allow or deny a certain request or the definition of a target in case of a defined delegation to a different service (e.g. an orchestrated service).

We make use of an open source Business Process Execution Language (BPEL)-based work-flow engine to incorporate more complex services that are the target for delegation as defined by a specific policy. This mechanism allows that the original exposed service interface remains the same from a third party service perspective. The executed service that will be called through a defined action as part of the policy nevertheless provides a specific business logic, defined by the operator policy for a specific (third party) service.

In order to simplify the generation of an executable service composition package visually composed with the Mash-up Studio, necessary design rules have been applied. A basic service is described via WSDL in Remote Procedure Call (RPC)/literal style and provides a single operation that only takes exactly one input message and exactly one output message. Within the XForms description some parts of the input message can be declared as configurable in the Graphical User Interface (GUI), while other parts are declared as input ports. All parts of the output message are declared as output ports respectively. Since the generated sequence of web-service invocations is control flow-oriented, the Mash-up Studio has to translate the dataflow-oriented composition properly to assure a meaningful execution order of the orchestrated basic services.

Based on the SDL description of the service SEE chooses adequate resources and controls the data flow in order to realise the service in the addressed network with high quality and performance. SEE uses standardised interfaces to access the basic services (WSDL, SOAP) and thus new services of third parties can easily be imported into the framework. Further on initial functions for monitoring and management of the running services are provided and will be extended in the future.

We have implemented a Parlay X-based network abstraction layer, the FOKUS OpenSE ([39] and [40]). Each Parlay X-based building block comprises operations for a specific telecommunications related functionality and consists of several operations that are available through a Web Service. It is represented by one or more services inside the Mash-up environment that provide the underlying enabler functionality in an abstract manner for service composition. The communication between the Mash-up Environment and the operator exposure layer (Parlay X gateway) is based on SOAP. As depicted in figure 5, each SOAP request towards the service provider / operator infrastructure is intercepted by the Service Broker.

The interfaces of the Service Broker are generic from a usage perspective meaning that the broker does not differentiate within its execution states between a service developer and an actual service being executed. We imposed a role-based system through applying different service scope policies that are evaluated by a Service Capability Manager.

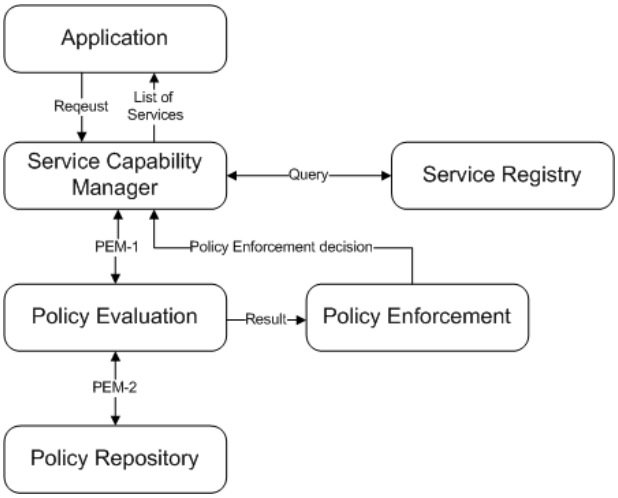

Figure 6 Service Capability Manager work-flow.

The Service Capability Manager allows the dynamic adaptation of services based on user, service or network/domain identifiers. A policy defines all services that are authorized for usage and mapped versus the available services at a service registry. An application may actively query the broker for allowed services through the PEM-1 interface to retrieve a list of services. Depending on the implementation of the application self-adaptation of the user interface is possible through this mechanism.

Services as well as developers get their usage scope associated through policy evaluation. A usage scope from a developer perspective results in a list of available services. Depending on the result of a Service Capability Manager query, the list of available services for the service developer is generated. Therefore a developer has only access to the services granted by the operator or service provider for service creation. Figure 6 provides a flow chart for this functionality.

The role of the service provider / operate can be considered as a super user role, that has access to all services, but furthermore also to the policy repository. A defined action as part of a policy may also include the invocation of a delegated service. Therefore, the SCE allows as part of the service provider role also the creation of delegated services that can be deployed as Web Services to the exposed platform. These services might either be directly exposed to third parties or being called within a policy for a specific other service to alter the service behavior depending on a specific policy.

5.3. Delivery Network

The 3GPP IP Multimedia Subsystem (IMS) provides the interfaces for interaction and underlying communication control infrastructure. The IP Multimedia Subsystem is defined from 3GPP Release 5 specifications on as overlay architecture on top of the 3GPP Packet Switched (PS) Core Network for the provision of real time multi-media services.

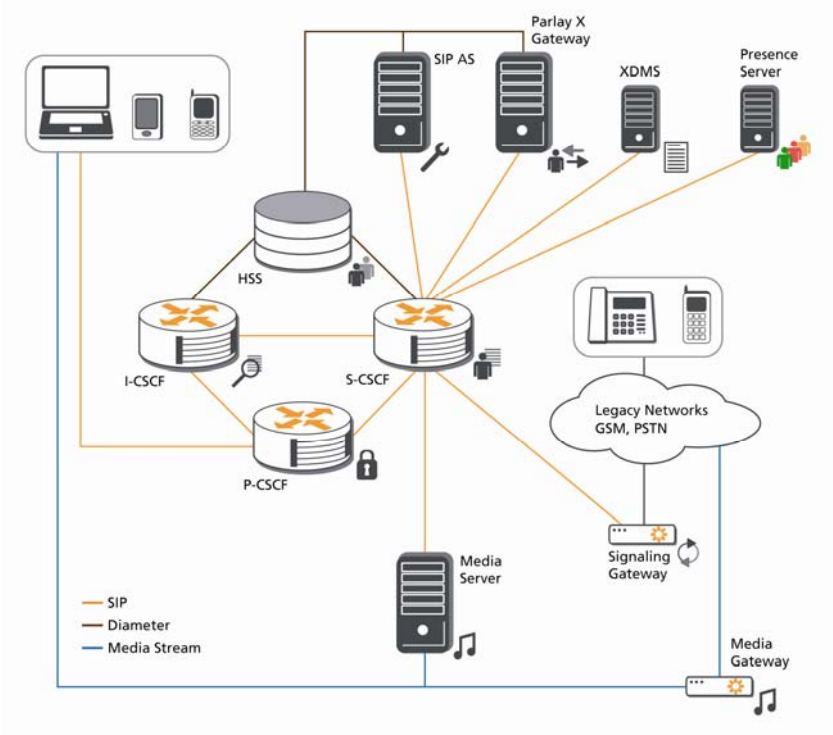

Figure 7 Basic IMS Architecture.

Due to the fact that the IMS overlay architecture is widely abstracted from the access network interfaces, it can be used for any mobile access network technology as well as for fixed line access technology as currently promoted by the European Telecommunications Standards Institute's (ETSI) Telecoms & Internet converged Services & Protocols for Advanced Networks (TISPAN) [41] within the Next Generation Network reference architecture definition.

The central session control protocols are the Session Initiation Protocol (SIP) and Diameter [42]. The SIP Application Server (AS) is the service relevant part in the IMS. It needs to support well

defined signalling and administration interfaces (3GPP ISC and Sh-interfaces) to connect to the standardized network architecture, consisting of the Home Subscriber System (HSS) storing subscriber related data and Call State Control Functions (CSCF) acting as soft switches for message routing and also providing security and authentication and authorization. Figure 7 depicts the simplified IMS architecture [43].

The particular techniques and methodologies that are required to gain the advantages of these key functionalities are not completely new, but the IMS provides the first major integration and the interaction of all key functionalities.

Therefore, a Network Abstraction Layer was introduced in-between the delivery network [44] and Service Execution Environment. OpenSE provides an implementation of the Parlay X interfaces. Each Parlay X-based building block (interface) comprises operations for a specific telecommunications related functionality and consists of several operations that are available through a Web Service that is mapped to a specific protocol of the IMS control layer.

6 Validation

We validate our service environment by benchmarking the execution of composed services and compare our approach and selection of technologies to an upcoming workflow expression language in standardization. We provide in this section a performance analysis for the following two candidates:

- BPEL
- State Chart XML

BPEL was chosen because of the widespread usage and the standard description language for the Web Service processes used by BPEL. State Chart XML (SCXML) [45] is able to describe complex state-machines; it is possible to describe notations such as sub-states, parallel states, synchronization, or concurrency. The objective of this standard is to formalize generic state diagrams notations, which are already used in other XML contexts as CCXML [46] or VoiceXML [47]. For the performance analysis between SCXML and BPEL the frequent orchestration of a simple SIP Instant Messaging service is assessed. The performance measurements are related to the service invocation:

a. in Java
b. with SCXML triggering a Java class
c. with SCXML triggering the service via Web Service
d. with BPEL (triggering also a Web Service)

The invoked service sends a simple SIP MESSAGE request to an application server that replies with a 200 OK. The implementation of the service is the same for every approach. For a) the service is simply executed directly from the Java method implementing the mentioned service. The second case b) has been invoked the Java method from inside the SCXML Engine. At last c) and d) need to call the Java method via Web Services from inside SCXML respectively the BPEL Engine.

In the experiment two physical servers are used. The first machine executes the service via the BPEL Engine (Apache ODE [48]), the SCXML Engine (Apache Commons SCXML [49]), the Web

Service and the Java implementation. Everything is deployed in a Jetty Web Server [50] in order to have comparable results.

The second machine deploys the SIP Message Servlet, which gets the SIP Messages and sends back a 200 OK response.

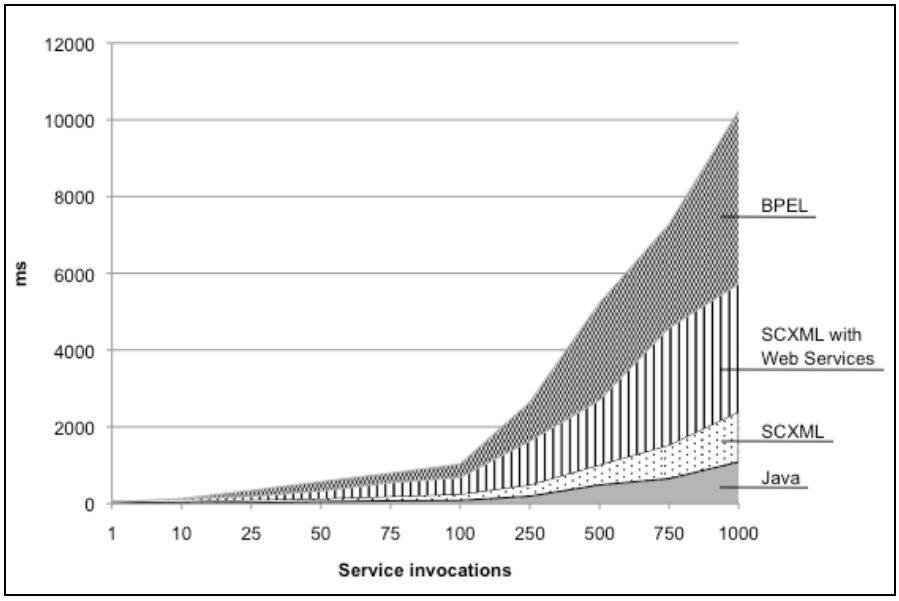

Figure 8 Performance of BPEL and SCXML.

The diagram in Figure 8 shows the time in milliseconds every approach needed to execute in relation to the frequency of service invocations. As expected the native Java access has the highest performance. We took Java into account of our measurements in order to see the computation overhead caused by the SCXML engine to execute the script in case b). In increased frequencies the execution time of SCXML is approximately 2.5 times higher than the native Java execution. Furthermore we wanted to get a picture of the overhead in using Web Services instead of native Java method calls in the SCXML engine. The performance decreases rapidly in case c) in particular for higher frequencies. The same applies for the last test case with BPEL. The orchestration overhead in SCXML is lower than the overhead in BPEL. Nevertheless this overhead in combination with Web Service calls brings out the lowest performance.

Nevertheless, our decision for choosing BPEL for the composition of workflows was also driven by the following factors:

- Mature support, tools, and runtime
- Existing homogeneous environment of Web Services

Therefore, a Java-only solution was not adequate despite the much better performance. As we plan to support in the future hybrid services consisting of multiple different service end points (e.g. REST, SOAP, JSON-RPC) we consider the migration to SCXML

7 Implementation of Show Cases

The service creation and execution environment was validated in a business area with partners bringing in concrete requirements and the relevant process expertise. We illustrate the functionality of our system with the help of two use cases. One topic is a commercial sports club representative for small and medium lifestyle-oriented companies, the other use case is targeted for after-care of patients after surgery.

7.1 Community Portal

Extending the communication for its mainly young club members, a social network has been established. We set up a community portal with relevant features and suitable layout of the sports club. Via the opensocial interface [51] the deployed services are exposed to the end users, i.e. the club members. We realized several aggregated services like the phone conference (see section 5), a video transmission from the clubs gym to the members mobile phones and desktops, a voicemail application, a date coordination service via SMS and email, and a online survey service with rich media charting.

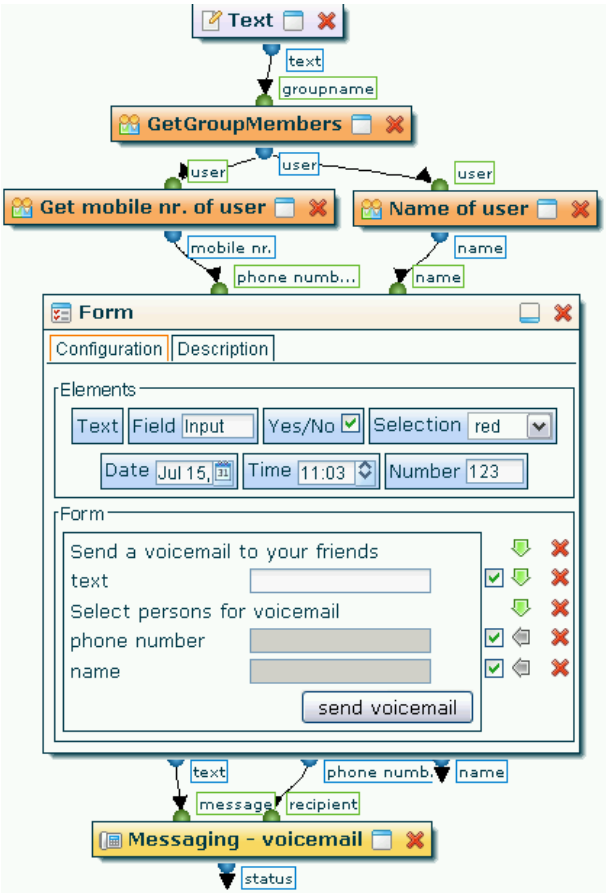

Figure 9 Service composition for voicemail service.

Figure 9 illustrates the service composition of the voicemail service. A basic service providing group members transmits member names to a web dialog form, in which during execution time the end user can select the members to receive the voice message and enter a text which is converted to speech.

Applying the service broker functionality, we changed the deployment of the phone conference service (figure 4) in a way, that dependent on the business model the conference call service is provided without or with a short advertisement before it is started. In case of advertisement, the service broker was configured with a policy which delegates the conference call to an additional service which concatenates an advertising speech in front of the conference call as an audio announcement (Parlay X Audio Call).

The video transmission service allows the clubs administrators to transmit special events from the sports center to interested club members. The devices of the users can be smart phones or internet connected PCs. At transmission time the clubs administrator signs on to the network with his camera. Via an IMS-Client a session is established to the media server to distribute the video signal to the devices of the invited members. The deployed web dialog is illustrated bellow in figure 10.

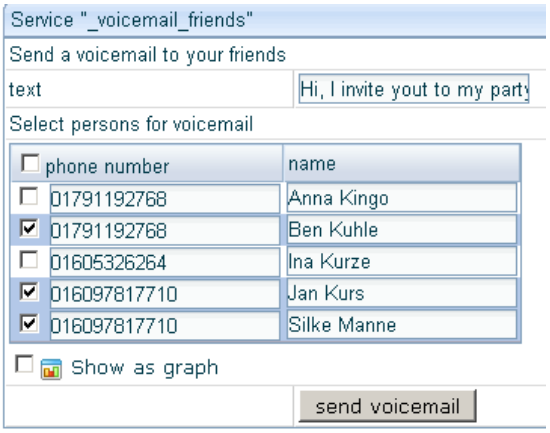

Figure 10 Deployed web dialog of voicemail service.

We use the service broker for further configuration activities like providing different basic services to service creators (see service capability manager), switch between different instantiations of enablers (e.g. SMS provided by FOKUS OpenSE or Deutsche Telekom developer garden).

7.2 After-care Clinic Support

Another application area in which we evaluated our mash-up environment was the aftercare of people with heart diseases. In Germany rehabilitation immediately subsequent to the treatment of an acute cardiovascular event is guaranteed by law. Rehabilitation lasts normally for 3 weeks either in a hospital or in an outpatient form. Various positive effects (blood pressure, significant increase in symptom-limited physical capacity, significant reduction in anxiety and depression) are observed after a 3-week rehabilitation treatment [52]. Observational studies show that the improved risk profile is not sustained over an extended period of time if the therapy is not consequentially continued in the subsequent care phase. To perform this aftercare programs hospitals are looking for opportunities to

support the patients at home. Telecommunication services provided by the hospitals like closed social networks, video conferences, phone conference, digital invitation, digital diaries are part of the user scenarios.

As example a digital diary set up by the rehabilitation expert is described in the following section.

The service composition uses the data in a dialog form (figure 11), saves it in a structured data store, which is realized per patient. The parameters to be stored in the diary are BMI (Body Mass Index), blood pressure (systolic and diastolic). Date and time are automatically included during runtime. In this example an alert should be sent to a relative of the person (e.g. daughter) in case that a threshold (e.g. blood pressure is greater 170) is reached. In this case the SMS service sends the alert to the phone number.

The diary input dialog presented in the health network is shown in figure 12. To view the diary another function allows the patient or a rehabilitation expert to view the results.

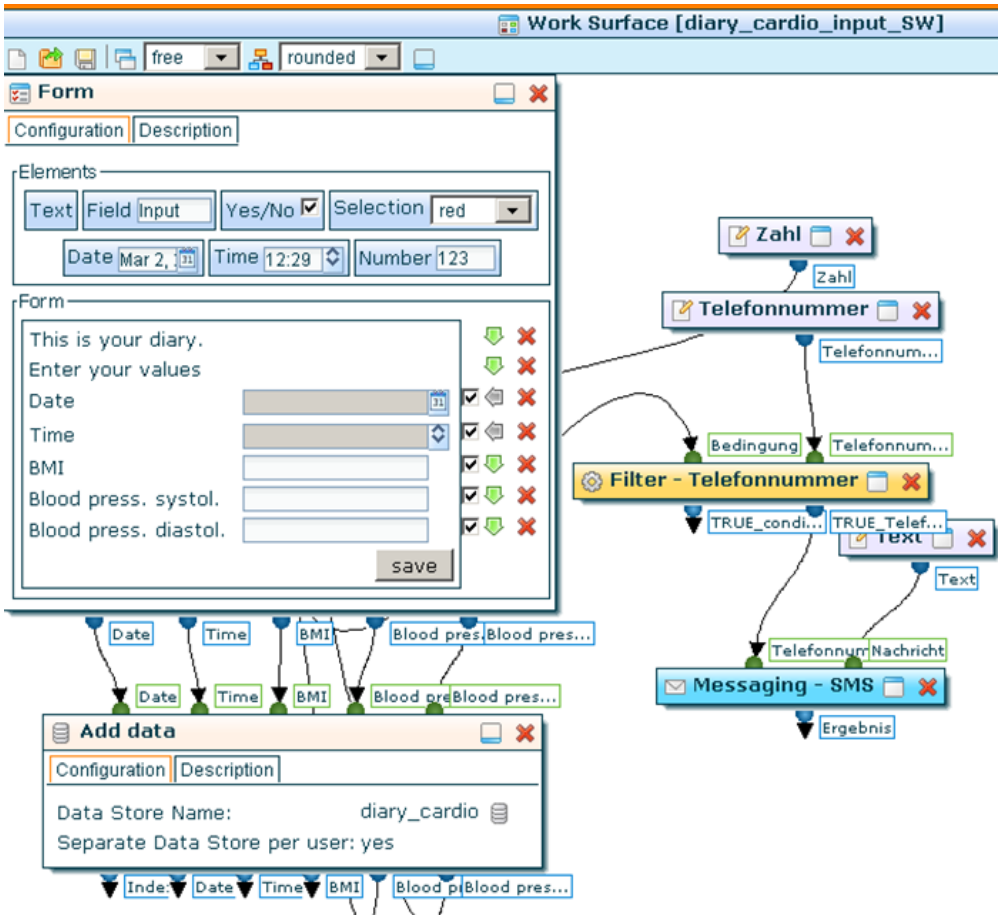

Figure 11 Service composition of aftercare diary (partly)

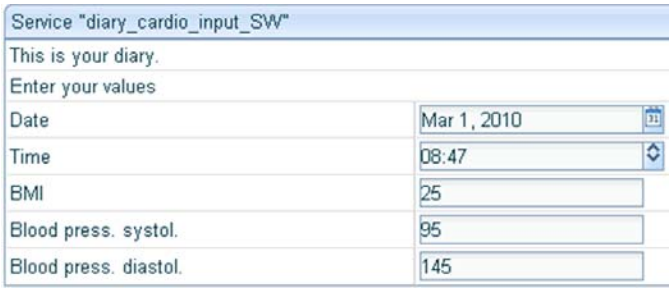

Figure 12: User input dialog to diary

Figure 13 shows the user dialog displaying the entries of the diary in a list form and a graphical form. We found that the principles of service creation and services exposure are usable in various application areas. The attractiveness of user generated services in company applications and the rehabilitation hospital scenarios was found during the work with the partners.

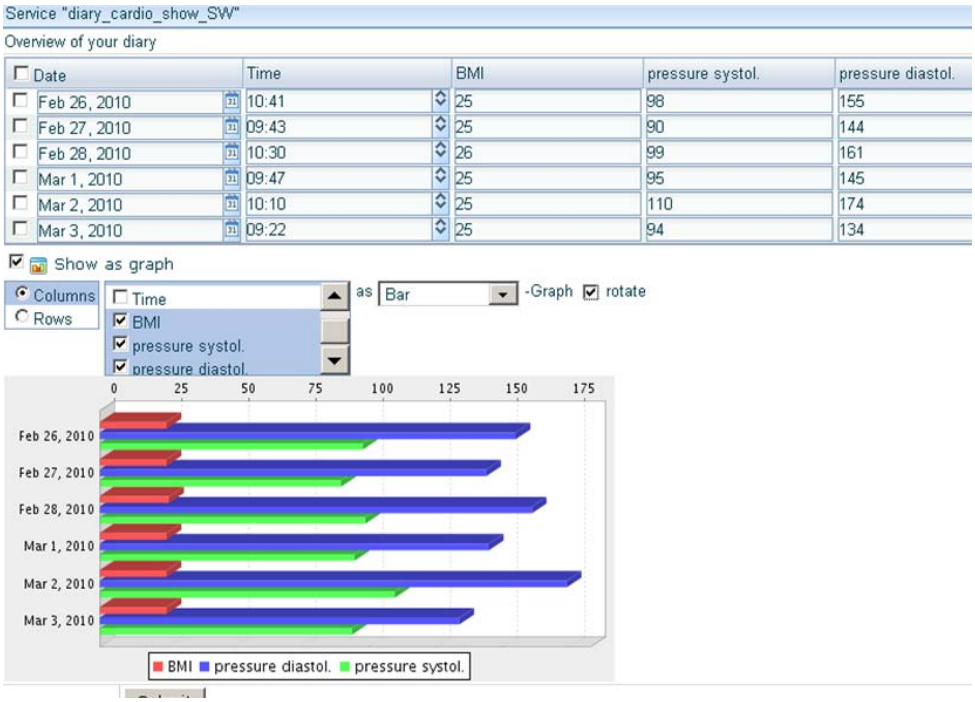

Figure 13 User disply dialog to diary

8 Conclusions and Outlook

SOA principles have been used inside telecommunications domains for many years, although different terms have been used over the last decades to describe the idea of realizing a programmable network to provide an open market of services. Today, Web Services-based APIs including emerging Web 2.0 interfaces represent the state of the art in SOA-based telecommunications, which are going to be integrated with the emerging NGN.

We have depicted in this paper our design for an open architecture that allows third parties and end users to create their own services with web-based graphical tool-kit. Service definition, access, and usage are controlled by a flexible service broker capable of defining fine-granulated service level agreements. The prototype system has been tested with several SMEs to include telecommunications specific services into a domain specific context; one specific service has been depicted in this paper.

Future steps will be performed to bring this system closer to production quality. The Mash-up Studio is supposed to act as an integrated component of an overarching service delivery platform for fixed and mobile networks for third party developer access. The Service Broker is under evaluation for becoming an active component of the developer garden.

Current short-comings of the system are the limitation to telecommunications related services and the missing openness of the system for user created service stubs of other services to be integrated into the Mash-up Studio.

Future academic work will concentrate on the dynamic service discovery, orchestration, and composition based on semantics, to allow the combination of many (operator-specific) service enablers automatically for more complex services and service assurance [53].

Acknowledgements

MAMSplus (Multi-Access, Modular-Services Framework) [34] is a joint project funded by the German Federal Ministry for Education and Research (BMBF) and managed by Deutsche Telekom Laboratories. Fraunhofer FOKUS and Technical University Berlin are partners within this project.

References

[1] Deutsche Telekom. http://www.developergarden.com.
[2] Ribbit. http://www.ribbit.com/.
[3] Orange Partner. http://www.orangepartner.com/.
[4] G. Finnie, Working with Third Party Services, “An Action Plan for Network Providers”, May 2009, Heavy Reading, http://downloads.lightreading.com/wplib/alcatellucent/WP_3rd_party_services.pdf
[5] Yahoo! Pipes. http://pipes.yahoo.com/pipes/.
[6] Lego Mindstroms. http://mindstorms.lego.com/.
[7] Scratch. http://scratch.mit.edu/.
[8] T. Magedanz, N. Blum, S. Dutkowski, “Evolution of SOA Concepts in Telecommunications - A Déjà vu?”, Special Issue on Service Oriented Architectures, IEEE Computer, November 2007, ISSN 0018-9162
[9] Parlay. http://www.parlay.org.
[10] Java Community Process. JSR 22: JAIN SLEE API Specification. http://jcp.org/en/jsr/detail?id=22
[11] H. Schulzrinne et al.. IETF RFC 3261. SIP: Session Initiation Protocol. 2002.
[12] 3GPP. TS 23.228. IP Multimedia Subsystem (IMS). Stage 2 v.7.10.0. 2007.
[13] Open Mobile Alliance (OMA). Enabler Release Definition for Push-to-talk over Cellular. Candidate Version 2.0 – 11 Dec 2007. 2007.
[14] Open Mobile Alliance (OMA). Presence SIMPLE Architecture Document. Approved Version 1.0.1 – 28 Nov 2006. 2006.

[15] Open Mobile Alliance (OMA). XML Document Management Architecture. Candidate Version 2.0 – 24 Jul 2007. 2007.
[16] Open Mobile Alliance (OMA). OMA Service Environment. Approved Version 1.0.4 - 01 Feb 2007. 2007.
[17] H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar, J. Polk, J. Rosenberg. RFC 4745 Common Policy: A Document Format for Expressing Privacy Preferences. February 2007.
[18] OASIS Web Services Business Process Execution Language Version 2.0. http://www.oasis-open.org/committees/wsbpel/. April 2007.
[19] Oracle White Paper—Oracle SOA Suite 11g. http://www.oracle.com/products/middleware/docs/microsite09-flashmedia-pdfs/getting-started-soa-suite-whitepaper.pdf
[20] Eclipsecon2009. Practical Process Orchestration using Eclipse SOA. http://www.eclipse.org/swordfish/assets/EC2009_BPEL.pdf.
[21] W3C Note. Web Services Definition Language (WSDL) 1.1. E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, March 15, 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315
[22] W3C Note. Simple Object Access Protocol (SOAP) 1.1. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, May 8, 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508
[23] OMG/BPMI. Business Process Modeling Notation, http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf
[24] L. Lin and P. Lin, “Orchestration in Web Services and real-time communications”, IEEE Commun. Mag. vol. 45. Jul. 2007. pp. 44-50.
[25] A. Jun, G. Kiss, H. Agrawal, P. Hyunseo, K. Sookyang, K. Sihn, and D. Kim, “An IMS-based service platform for the next-generation wireless networks”, IEEE Commun. Mag. vol. 44. Sept. 2006. pp. 88-95.
[26] S. Q. Khan, R. Gaglianello, and M. Luna, “Experiences with blending HTTP, RTSP, and IMS”, IEEE Commun. Mag. vol. 45. Mar. 2007. pp. 122-128.
[27] R. Levenshteyn and I. Fikouras, “Mobile services interworking for IMS and XML web services”, IEEE Commun. Mag. vol. 44. Sept. 2006. pp. 80-87.
[28] Xu Shao, Teck Yoong Chai, Teck Kiong Lee, Lek Heng Ngoh, Luying Zhou, Markus Kirchberg, "An Integrated Telecom and IT Service Delivery Platform", APSCC. pp.391-396. 2008 IEEE Asia-Pacific Services Computing Conference. 2008.
[29] R.H. Glitho; F. Khendek; A. De Marco, "Creating value added services in Internet telephony: an overview and a case study on a high-level service creation environment," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.33, no.4, pp.446-457, Nov. 2003
[30] N. Blum, T. Magedanz, J. Kleeßen, T. Margaria, "Enabling eXtreme Model Driven Design of Parlay X-based Communications Services for End-to-End Multiplatform Service Orchestrations," Proc. of 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), pp.240-247, 2009, ISBN 978-0-7695-3702-3
[31] N. Blum, T. Magedanz, H. Stein, “Service Creation & Delivery for SME based on SOA / IMS”, MNCNA ’07, Nov. 26, 2007 Port Beach – CA, ISBN 978-1-59593-932-6
[32] FIRE - Future Internet Research & Experimentation, http://cordis.europa.eu/fp7/ict/fire/
[33] C. Baladron; J. Aquiar; B. Carro; L.W. Goix; A. Leon Martin; P. Falcarin; J. Sienel, “User-Centric Future Internet and Telecommunication Services”, Future of the Internet Conference, Prague (Czech Rep.) May 2009.
[34] Multi Access, Modular-Services Framework, BMBF http://www.mams-platform.net/.

[35] White Book. Deutsche Telekom AG Pattern Language - Consistent Interaction Logic for Services & Applications by Deutsche Telekom Laboratories. 1. Imprint 2007.
[36] Open Mobile Alliance (OMA).Policy Evaluation, Enforcement and Management Callable Interface (PEM1). 2008. http://www.openmobilealliance.org.
[37] Open Mobile Alliance (OMA).Policy Evaluation, Enforcement and Management – Management Interface (PEM2). 2008. http://www.openmobilealliance.org.
[38] N. Blum, I. Boldea, T. Magedanz, U. Staiger, H. Stein, "A Service Broker providing Real-time Telecommunications Services for 3rd Party Services", Proc. of 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC), Seattle, July 2009, ISBN 978-0-7695-3726-9, DOI 10.1109/COMPSAC.2009.202.
[39] FOKUS OpenSE, www.opensoatelcoplayground.org/opense
[40] F. Schreiner, S. Wahle, N. Blum, T. Magedanz, "Modular Exposure of Next Generation Network Services to Enterprises and Testbed Federations", HUT-ICCE 2008, 2nd International Conference on Communications and Electronics, June 4-6, 2008, Hoi An, Vietnam, ISBN 978-1-4244-2425-2, IEEE CN CFP0816B-PRT
[41] ETSI. http://www.etsi.org/tispan/.
[42] P. Calhoun. IETF RFC3588. Diameter Base Protocol. 2003.
[43] D. Vingarzan, P. Weik, T. Magedanz, "Development of an Open Source IMS Core for Emerging IMS Testbeds", The Academia and Beyond, Journal of Multimedia, Vol. 0, No. 0 (2005) 000 000 © Rinton Press, ISSN: 1550-4646
[44] Open SOA Telco Playground, www.opensoaplayground.org.
[45] W3C, "State Chart XML (SCXML): State Machine Notation for Control Abstraction", 2009, http://www.w3.org/TR/scxml/
[46] W3C, "Voice Browser Call Control: CCXML Version 1.0", 2007, http://www.w3.org/TR/ccxml/
[47] W3C, "Voice Extensible Markup Language (VoiceXML) Version 2.0", 2004, http://www.w3.org/TR/voicexml20/
[48] Apache ODE - http://ode.apache.org/
[49] Apache Commons SCXML -http://commons.apache.org/scxml/
[50] Jetty Web Server - http://jetty.codehaus.org/jetty/
[51] Opensocial. http://code.google.com/apis/opensocial.
[52] Karoff, M., Held, K., Bjarnason-Wehrens, B., "Cardiac rehabilitation in Germany", The European Society of Cardiology, 2006
[53] N. Blum, T. Magedanz, F. Schreiner, "Management of SOA based NGN service exposure, service discovery and service composition", Proc. of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), New York, USA, 1 - 5 June 2009.