Personalized Service Creation and Provision for the Mobile Web

Quan Z. Sheng and Jian Yu and José M. del Álamo and Paolo Falcarin

Abstract The convergence of telecom networks and the Internet is fostering the emergence of environments where Web services are available to mobile users. The variability in computing resources, display terminal, and communication channel require intelligent support on personalized delivery of relevant data and services to mobile users. Personalized service provisioning presents several research challenges on context information management, service creation, and inherent limitations of mobile devices. In this chapter, we describe a novel framework that supports weaving context information and services for personalized service creation and execution. By leveraging technologies on Web services, agents, and publish/subscribe systems, our framework enables an effective, user-centric access of integrated services over the mobile Web environments. This chapter overviews the design, architecture, and implementation of the framework.

Quan Z. Sheng
School of Computer Science, The University of Adelaide, Australia, e-mail: qsheng@cs.adelaide.edu.au

Jian Yu
School of Computer Science, The University of Adelaide, Australia, e-mail: jyu01@adelaide.edu.au

José M. del Álamo
Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain, e-mail: jmdela@dit.upm.es

Paolo Falcarin
Department of Control and Computing Engineering, Politecnico di Torino, Italy, e-mail: paolo.falcarin@polito.it

1 Introduction

The advances of telecom and mobile technologies are changing the way services are delivered to end users. With the convergence of telecom networks and an all-Internet Protocol (IP) based core network—on the one hand—typical telecom services like voice call and Short Message Service (SMS) become accessible to Internet users—on the other hand—mobile users are also granted the opportunity to enjoy versatile Internet services from anywhere and at anytime [9, 4, 31]. In this sense from the end user point of view, mobile Web means using mobile devices to access both telecom and Internet services in a homogeneous way. Services become more accessible to end users. In addition, location and presence based service personalization is destined to provide richer and finer user experiences [5, 23].

Recently, a new paradigm has emerged allowing non-skilled end users to easily generate their own content and services from the mashup of existing information services and data sources [7]. Web 2.0 [29] is the name given to this paradigm, which includes new communication channels like blogs or wikis. They support end users to communicate following a many-to-many scheme. This scheme is usually represented by a virtual community, where people with common interests create and share their own content. The Web 2.0 paradigm and virtual communities provide the basis to enhance the existing communications model with people, thus supporting weaving data, services and people over the Web. Recognizing this trend, telecom communities have realized that they can add a new ingredient: location to Web 2.0. Location, as any kind of context information, is a feature that allows a new generation of active communications services supported by rich service personalization and enhanced end user experience. These services are able to properly react according to end user, device or service context information [4, 5]. However, end user location information is usually difficult (if not impossible) to obtain on the Web. The extension of the Web to mobile devices, or so called *mobile Web*, is a must in order to succeed in this direction [23].

In order to create and provide/deliver personalized services for the mobile Web, we need to answer several key research questions in the first place. How to retrieve and manage context information from various context providers is at the very core of these issues. Secondly, apart from rendering rich and intuitive graphical service creation environment, non-expert users need considerable help in making a sound composite service. Some inherent complexity of programming should be shielded from these users. Last but not the least, we need to handle the short-of-resource feature of mobile devices. Mobile devices posses, to a certain extent, limited resources (e.g., battery power and input capabilities). Therefore, mobile devices should better act as passive listeners (e.g., receiving the results) than as active tools for service invocation, so that the computational power and battery life of the devices can be extended.

In this chapter, we introduce an innovative framework that supports weaving distributed data, context information, and communication services to allow personalized service creation and provision for mobile users. In this multi-agent based framework, user context is acquired by integrating presence and location service en-

ablers provided by the underlying telecommunications network. A template-based service creation environment allows end users to build personalized integrated services by simply configuring a process template with personal information. Finally, publish/subscribe techniques are used to ensure the passiveness of client devices. It is worth noting that agent technology help greatly in improving the adaptability of this framework.

The structure of this chapter is organized as follows. In Section 2, we briefly introduce the trends of how the World Wide Web is moving towards the mobile Web, together with a motivating example. In Section 3, we discuss the design principles of our framework, including our ideas on how to retrieve context information, the service creation approach, and the service provision approach. The whole system architecture of the framework is described in Section 4. Section 5 reports the implementation status and the initial evaluation. Section 6 overviews the related work and finally Section 7 concludes the chapter.

2 Towards the Mobile Web

During the last few years, there has been a convergence process in the telecommunications area, driven by three main axes: device convergence, multimedia convergence and network convergence. Devices have converged in the sense that now we can use a single device to do the things that we needed different devices in the past such as television, computer, and telephone. Multimedia has converged in a way that we can have voice, data and video in a single service.

Telecom network architectures have evolved from the original Public Switched Telephone Network (PSTN). Nowadays several access and core networks coexist both in wireless and wired-line domains. The evolution process has been characterized first by the introduction of mobile technologies in the early 1990s, and the generalized access to the Internet later. As a result of this process several access networks have been deployed. Just to mention a few, there are fixed networks such as the aforementioned PSTN and the Integrated Services Digital Network (ISDN), several radio networks such as the Global System for Mobile communications (GSM), the General Packet Radio Service (GPRS), the Universal Mobile Telecommunications System (UMTS) and the CDMA2000 networks (Code Division Multiple Access - CDMA), other wireless technologies such as WiFi and WiMax. Core networks have also evolved from the original circuit switched networks to the packet switched networks. At the beginning each network developed its own protocols with the aim of enhancing the previous ones. However, that means the coexistence of several network protocols that should communicate with each other, which produces both technological and business problems. Besides, each network followed a vertical approach where access entities, control systems and services were tightly coupled.

2.1 Expanding the World Wide Web to the Mobile Web

The convergence process for all the different networks began in 1998 when leading regional telecommunication standard bodies agreed on creating two 3rd Generation Partnership Projects (3GPP[1] and 3GPP2[2]), aiming at evolving the circuit and voice based mobile networks into 3rd Generation (3G) multimedia packet-based mobile networks. The main result of these projects was the specification of the IP Multimedia Subsystem (IMS) [9].

IMS is an architectural framework for delivering IP multimedia services to end users. It was originally focused on evolving mobile networks beyond GSM and thus its initial formulation represented an approach to delivering Internet services over GPRS. This initial approach was subsequently updated by 3GPP itself, 3GPP2, the Telecommunication and Internet converged Services and Protocols for Advanced Networking (TISPAN) and the Alliance for Telecommunication Industry Solutions (ATIS) by providing requirements for support of packet-based networks other than GPRS such as Wireless Local Access Networks (WLAN), CDMA2000 and fixed line. Finally, due to concerns on possible overlaps, delays, and incompatibilities among future standards, the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T)[3] has coordinated the standardization process.

As a result, nowadays there is a common IP core network with multiple IP-based access networks. Next Generation Network (NGN) is the term used to refer to these new IP-based network architectures. The ITU-T has defined an NGN as “a packet-based network able to provide telecommunication services and able to make use of multiple broadband, Quality of Service (QoS)-enabled transport technologies and in which service-related functions are independent from underlying transport-related technologies. It enables unfettered access for users to networks and to competing service providers and/or services of their choice. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users”.

NGN architectures allow for having services that were previously precluded to mobile networks due to signaling delays and latencies, low throughput, etc. End users now can have personal broadband mobile services at any time and any place and can begin to expect the flexibility, scope, and service variety they have experienced through the Web: therefore the raise of the mobile Web.

This new IP, multimedia-based mobile Web has blurred the border between the Internet and Telecommunications domains. Moreover, as regulation has required Telecommunications operators to open up their networks to third parties development, this has allowed new competition to be introduced into the telecom service provision business, and nowadays it is possible for third parties outside the network operator domain to develop new services for the Web enhanced with mobile features.

[1] http://www.3gpp.org.

[2] http://www.3gpp2.org.

[3] http://www.itu.int/ITU-T.

2.2 Opening Up the Mobile Web for Development

The efforts to open up Telecommunications networks can be initially dated as early as the 1980s with the standardization of Intelligent Networks (IN) [17]. IN is an architecture by which customized service logic programs can be created by a service provider for enhanced features on calls in the PSTN.

IN introduced a set of functional entities comprising the distributed functions that need to interact during call origination and call termination in the provision of IN call-related services, thus decoupling the service development from the network infrastructure. From that, the IN infrastructure has evolved to support some of the new features and requirements of the evolving networks. For instance CAMEL, which is the technology used in GSM networks to enable services such as roaming and international pre-paid, is based on IN [42].

However, IN is not able to fulfill some of the requirements that new converged networks impose such as shorter time to market of new services and the development of value-added, network independent services. To cover this gap IP-based NGN architectures such as the IMS have been specified. Initiatives such as the Open Service Architecture (OSA), Parlay, the Open Mobile Alliance (OMA)[4], and the JAIN Service Logic Execution Environment (JAIN SLEE or JSLEE) [27] have also been developed in different context.

The specification of the OSA/Parlay is a joint effort between ETSI, 3GPP and the Parlay Group[5]. It is an open Application Programming Interface (API) for application access to telecoms network resources. OSA/Parlay technology integrates telecom network capabilities with IT applications via a secure, measured, and billable interface. The APIs are network independent, and applications can be hosted within the telecom network operator's own environment, or by external third party service providers.

A Web services API is also available, known as Parlay X[6]. This is also standardized jointly by ETSI, Parlay and the 3GPP. Parlay X is a simplified Web services interface to telecom network functionality. It may be used in conjunction with the base OSA/Parlay APIs, or on its own. Within the mobile domain, OMA is the focal point for the development of service enabler specifications. OMA does not deliver platform implementations, but provides specifications of service enabling functionalities such as location, instant messaging, and device management. It is left to others to implement the platforms and functionalities that they describe, specially using Web services technologies.

JSLEE [27] defines a component model for structuring application logic of communications applications as a collection of reusable object-orientated components, and for composing these components into higher level, richer services. The specification also defines the contract between these components and the container that will host these components at runtime. JSLEE provides support asynchronous ap-

[4] http://www.openmobilealliance.org.

[5] http://www.parlay.org.

[6] http://www.parlayx.com/.

plications based on an event model. Application components receive events from event channels that are established at runtime. Network resource adapters create representations of calls and pass events generated by the calls to the JSLEE. Application components are in turn invoked by the JSLEE to process these events in a transactional context.

2.3 A Motivating Scenario

Let's assume a university student, Donatella, wants to use her PDA as an advanced campus assistant to help organize her daily life efficiently. The following describes some most significant items extracted from her wish list.

First of all, although an engineering student, she is also interested in history and arts, especially art history and any paintings, sculpture and architecture from ancient time. She hopes that her PDA could notify her whenever a lecture of interest is posted on the university's website. Furthermore, she hopes that how the PDA gives notifications could base on her presence: when attending a class, she would like to be informed only after class; when riding a bicycle or driving a car, she would like the PDA give her a voice message; and at any other situations, she is pleased if the notification arrives as a piece of short message. She hopes that her presence could be changed intelligently based on her calendar and location parameters. In other words, at anytime when her calendar is marked as a lecture and she is within one of the campus buildings, her presence will be set to `Attending Lecture`. Whenever she is moving at an average speed higher than 35KM/H, her presence will be set to `Driving`. Of course she can manually override the automatic settings.

During the lecture, she wants to submit her questions via the PDA. She browses the questions asked by other students and decides to either vote for a posted question (if a similar question has been already asked), or post her question (if no one has asked yet). She may ask several questions during the lecture. After the class, a consultation booking might be requested and a feedback on the lecture is provided by her.

3 Design Principles

In this section, we describe the main design principles adopted in creating an personalized service creation and provision architecture. These design principles target the major challenges in the mobile Web field we mentioned at the beginning of this chapter.

3.1 Service Provision Approach

Our approach aims at identifying the main abstractions and mechanisms that support service provisioning in the mobile Web. We believe that leveraging Web services and software agents in combination with publish/subscribe systems provides the foundation to enable effective access to integrated services in mobile Web environments.

Web Services. Web services provide the pillars for evolving the Internet into a service-oriented integration platform of unprecedented scale and agility. The foundation of this platform lies in the modularization and virtualization of system functions and resources as services that: (i) can be described, advertized and discovered using (XML-based) standard languages, and (ii) interact through standard Internet protocols. More specifically, Web service technology is characterized by two aspects that are relevant to accessing heterogeneous resources and applications. The first is that from a technology perspective, all interacting entities are represented as services, whether they providing or requesting services. This allows uniformity that is needed for the interaction among heterogeneous applications and resources such as mobile devices. Web services can bring about the convergence of wired and wireless applications and services. Since Web services are described and interacted in a standardized manner, the task of developing complex applications by composing other services is considerably simplified [30, 40]. This allows for the aggregation of resources and applications for completing a specific complicated task.

Agents. Agents are software entities that exhibit certain autonomy when interacting with other entities [41]. Agents use their internal policies and knowledge to decide when to take actions that are needed to realize a specific goal. Internal policies can use context information (e.g., user preferences, device characteristics, and user location) to enable agents to adapt to different computing and user activities. Agents can also be used to pro-actively perform actions in dynamic environments. In fact, the combination of services and agents, will provide a self-managing infrastructure. Agents extend services by embedding extensible knowledge and capabilities (e.g., context aware execution and exception handling policies) making them capable of providing personalized and adaptive service provisioning in dynamic environments.

Publish/Subscribe System. The publish/subscribe paradigm offers a communication infrastructure where senders and receivers of messages interact by producing and consuming messages via designated shared spaces [11]. The communication is *asynchronous* in the sense that it completely decouples the senders and receivers in both space and time. This enables mobile users to disconnect at any time—either voluntarily to save e.g., communication cost and battery power, or involuntarily due to breakdowns of connections—and re-synchronize with the underlying infrastructure upon reconnection. This form of decoupling is of paramount importance in mobile Web environments, where the entities (e.g., mobile users) involved in communication change frequently due to their movement or connectivity patterns. This communication paradigm caters for loosely coupled interactions among ser-

vice agents, which has been identified as an ideal platform for a variety of Internet applications, especially in wireless environments.

A widely recognized effort is the *tuple space model* that has its origins in the Linda language [2] for distributed programming. Tuple spaces have the advantage of providing direct support for *pull-based asynchronous interactions* in which the "sender" (e.g. the client device) and the "receiver" (e.g. a component service) are separated in both space and time. This enables mobile users to disconnect at any time, and re-synchronize with the underlying infrastructure upon reconnection. Tuple spaces also support multi-channel interactions, since a given user can interact with a tuple space from multiple computing resources (e.g. place a tuple from a device, then disconnect, and then reconnect and get a tuple from another device).

3.2 Context Information Retrieval

In order to personalize the service creation and provision processes some context information is needed. This information can be categorized, within the scope of this work, into three major sets: *user context*, *device context*, and *service context*.

User Context. User context refers to the information related to a user such as preferences, calendar or location. This information is also referred to in the literature as *Personal Identifiable Information* (PII). PII is any piece of information that—related both digital and real identities—that can potentially be used to uniquely identify, contact, or locate a single person [10, 37].

Telecom operators have gathered loads of information about their customers over the years, thus creating rich users' profiles. They are also able to retrieve dynamic personal information such as customers' location or presence status. Therefore, we leverage on the underlying telecom infrastructure to get user context information. This information is exposed by means of Web services interfaces, which are standardized as OMA enablers.

Device Context. The device context information includes hardware and software characteristics of the user's devices, which could be used to drive the service execution or tailor the contents that a service will present to the user.

The World Wide Web Consortium (W3C) has created the Composite Capability/Preference Profiles (CC/PP) Recommendation [15], which allows defining capabilities and preferences of users' mobile devices. OMA has specified the *User Agent Profile* (UAProf) [38], capturing capabilities and preference information for wireless devices. UAProf allows describing device and user agent capabilities as an XML description, and is fully compatible with CC/PP. Moreover, OMA has also specified the Mobile Client Environment enabler, which is specifically chartered to be responsible for base content types, with the intention of enabling the creation and use of data services on mobile hand held devices, including mobile telephones.

Service Context. The service context includes information related to a service. Examples of service context include: i) service location, ii) service status (e.g., avail-

able, busy), and iii) Quality of Service (QoS) attributes (e.g., price, availability). The service context information can be obtained via a monitoring application that oversees the execution status of the service (e.g., exceptions). For example, if the value of a monitored item (e.g., CPU usage) is beyond a critical point (e.g., 85%), the workload of the service will be set as `heavy`.

Context Privacy and Security. The Internet experience has demonstrated that personal data (i.e., user context information) is a valuable asset that can be used incorrectly or fraudulently, and thereafter is prone to be abused and misused. Fortunately, in most countries there are laws that require companies to ensure security and privacy when using, storing or revealing personal information about a customer [19].

In the technical plane, Identity Management (IM) is the discipline that tries to address all these issues: it refers to the processes involved with management and selective disclosure of PII within an institution or between several of them, while preserving and enforcing privacy and security needs. If different companies have established trust relationships between themselves the process does not require a common root authority and is called *Federated Identity Management*. Each company maintains its own customer list and some identity attributes or PII, but they can securely exchange them while preserving users' privacy. IM is related to network security, service provisioning, customer management, Single Sign On (SSO), Single Logout (SLO) and the means to share identity information [39].

A federated approach for IM in a Web Services environment is supported by various standards and frameworks, among which the most important ones are Security Assertion Markup Language (SAML)[7], Liberty Alliance[8], and WS-Federation [16]. While all of them support similar features, Liberty Alliance specifications have got greater acceptance within the telecom domain.

3.3 Service Creation Approach

In a world where technology complexity is growing, one of the big and constant challenges is to keep the technology reachable by users while increasing the amount of features available. This is even more critical in recent trends of Web development, where the buzzword "Web 2.0" [29] represents, along with a new rich set of Web technologies, the goal of bringing a user from a passive role to an active role in the system, both as a content producer and as a service creator. Sites as Wikipedia, Flickr, YouTube, or the blog phenomenon show how the user is now assuming the content producer role too, providing expert knowledge and information, images, and video clips.

One step more is needed to allow users to create not only static content, but applications and services. The success of this trend is proved by the increasing popularity of mashups (i.e., little applications that combine information from many web sites)

[7] http://docs.oasis-open.org/security/saml/v2.0.

[8] http://www.projectliberty.org.

and the birth of various environments for the intuitive non-expert creation of Web based information services, driven by the biggest and most successful companies of the IT world, such as Yahoo! Pipes[9] or Microsoft Popfly[10]. These environments present graphical tools based on drag-and-drop interfaces that allow a user to create Web-based applications even without any computing knowledge.

In the same direction, the OPUCE (Open Platform for User-centric Service Creation and Execution) platform[11] aims at bridging advances in networking, communication and information technologies towards a unique converged service creation environment, where personalized IT-telecom services can be created by end users. Another related approach is provided by the SPICE service creation environment [4], where natural-language goals are interpreted by an Automatic Composition Engine (ACE) in order to obtain a set of suitable compositions of semantic-annotated services [34]. Such compositions can be evaluated and possibly deployed by a service designer and its GUI is intuitive enough to be used by an end user.

The challenge for all these research initiatives is to put a non-expert in the center of a Service Creation Environment (SCE). Some approaches based on rules (like ECA rules) [25] or policy languages [24] have been used in telecom world for personalization of value-added services from the service developer's viewpoint, as long as the usage of scripting languages, and domain-specific SCEs [26].

Our personalized service creation approach is centered on the concept of *process templates*. Users specify their needs by reusing and adjusting existing process templates, rather than building their own services from scratch.

Process Templates. Process templates are reusable business process skeletons that are devised to reach particular goals such as arrangement of a trip. Each process template has one or more tasks and each task is associated with a service operation. Process templates are modeled using process definition languages such as statecharts [20] and BPEL4WS [3].

In a process template, each task has a set of input and output parameters. The value of a task's input parameter may be: i) requested from user during task execution, ii) obtained from the user's profile, or iii) obtained as an output of another task. For the second and third cases, the value of an input parameter is obtained via queries. Queries vary from simple to complex, depending on the application domain and users' needs. Queries can be expressed using languages like XPath [13].

In our approach, values that users supply as input parameters are handled differently from the values obtained from user profiles. Indeed, because mobile devices are resource-constrained, values that can be obtained from user profiles should not be requested from users. Users only supply the values for data items that are labeled *compulsory*. However, in a process template specification, the template provider only indicates which input parameters users have to supply a value. It is the responsibility of the user to specify, during the configuration phase, if the value will be provided manually or derived from her profile.

[9] http://pipes.yahoo.com.

[10] http://www.popfly.ms.

[11] http://www.opuce.eu.

Similarly, the value of a task's output parameter may be: i) sent to other tasks as input parameters, and/or ii) sent to a user in case she wants to know the execution result of the task. Note that the value of an output parameter can be submitted to multiple places (e.g., to a task and the user as well). Again, due to low bandwidth and limited presentation capabilities of certain mobile devices, results of tasks are progressively delivered to users. Similar to input parameters, the provider of a process template does not decide which output parameters need to be returned.

Configuring Process Templates. Personalization implies making adjustment according to user preferences. Three kinds of user preferences are associated for each process template's task:

- *execution constraints* can be divided into *temporal* and *spatial* constraints, which respectively indicate *when* and *where* the user would like to see a task executed,
- *data supply and delivery preferences* are related to supplying values to the input parameters and delivering values of output parameters of the task, and
- *execution policies* are related to the preferences on service selection (for communities) and service migration during the execution of a task.

Generally, a temporal constraint involves current time, *ct*, comparison operator, *co* (e.g., =, $\leq$, and `between`), and a user-specified time, *ut*, which is either an absolute time, a relative time (e.g., termination time of a task), or a time interval. A temporal constraint means that a task can be triggered only if the condition *ct co ut* is evaluated to true. Similarly, a spatial constraint involves current location, *cl* and a user-specified location, *ul*. A spatial constraint means that a task can be fired only when the condition $cl = ul$ is evaluated to true. A location is considered the same as another location if the distance between two locations does not exceed a certain value (e.g., 2 meters). It should be noted that the temporal and spatial constraints can be empty, meaning that the corresponding task can be executed at anytime and at anywhere.

As stated before, the values of some input parameters of a task can be obtained from a user's profile. The user proceeds in two steps: i) identify which input parameter values can be derived from her profile, and ii) supply the location of the profile and the corresponding attribute names. Similarly, for the output parameters of a task, a user may specify which parameter values need to be delivered to her.

The execution policies include the *service selection policy* and the *service migration policy*. For a specific task, users can specify how to select a service for this task. The service can be a fixed one (the task always uses this service), or can be selected from a specific service community [6] or a public directory (e.g., UDDI) based on certain criteria (e.g., location of the mobile user). Furthermore, users can specify whether to migrate the services to their mobile devices (e.g., if mobile devices have enough computing resources) or to the sites near the users current location for the execution. Policies can be specified using policy languages like Houdini [21] and Ponder [28].

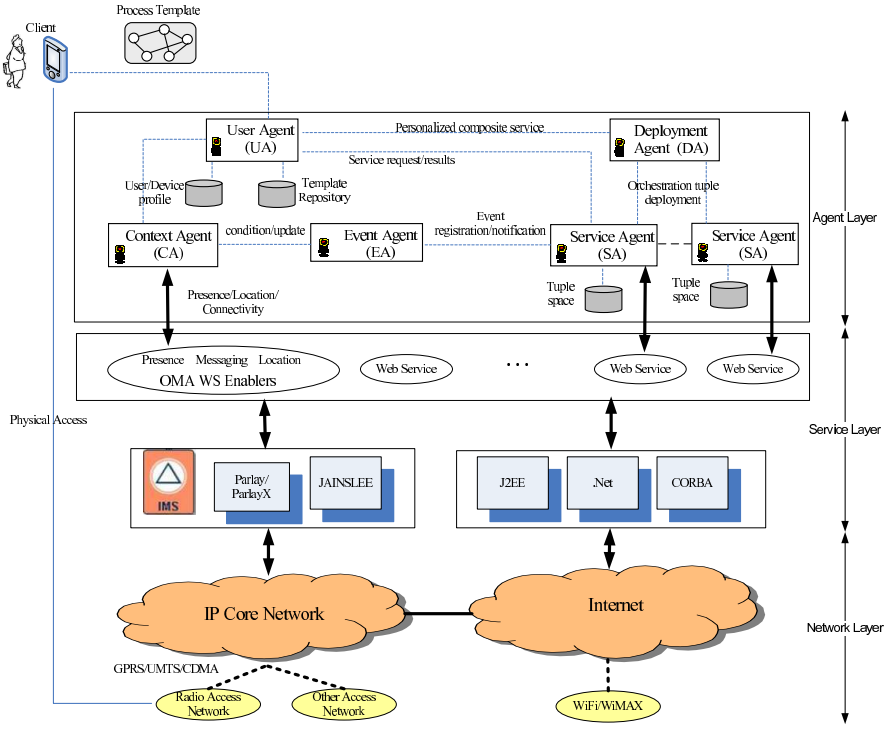


Fig. 1 System architecture

4 System Architecture

In this section, we introduce the architecture of our context-aware service creation and execution framework. As shown in Figure 1, the framework is separated into several logical layers, namely the *agent layer*, the *service layer*, and the *network layer*.

4.1 Agent Layer

The agent layer consists of a set of agents that collaborates among each other for the robust and context-aware creation and execution of composite services in the mobile Web environments.

User Agent. Mobile users access our service provisioning environment through two main components, namely the *client* and the *user agent*. The client is an application that can be downloaded and runs on mobile devices. It provides users with an interface for i) specifying user activities, and ii) interacting with the user agent.

Users' activities (e.g., travel planning) are usually complex. The fulfillment of an activity may call for multiple services executed in a specific chronology. It is too tedious to specify activities from scratch through small devices like PDA, which have limited input capabilities. To ease the activity specification process, we introduce the notion of *process templates*. We specify process templates with statecharts [20]. Succinctly, a statechart is made up of states and transitions. States can be basic or compound. A basic state (also called task) corresponds to the execution of a Web service. Compound states enclose one or several statecharts within them.

The client allows users to define their activities by specifying *personal preferences* (e.g., temporal/spatial constraints, data supply/delivery preferences) over the tasks of process templates, thereby defining *personalized composite services*. A user can specify *temporal* and *spatial* constraints for each task, which respectively indicate *when* and *where* the user wants to have a task executed. For example, Donatella, the university student in our motivating example (see Section 2.3), can specify that the question asking services (e.g., `Question Vote` service) can be executed only during the lecture time and when she is in the classroom. Meanwhile, data supply and delivery preferences are related to supplying values to the input parameters and delivering the values of output parameters of a task. Donatella can specify that the value of a task's input parameter should be obtained from her profile so that she does not have to provide the value manually. Similarly, she can also specify that the value of a task's output parameter should be sent to her user agent in case she wants to know the results.

A user agent (UA) acts on behalf of a specific mobile user. The UA maintains profile information including the user's contact information, personal preferences, and mobile device characteristics. Based on the profile, the UA can identify data formats and interaction protocols for communicating with the client that runs on the mobile device. The UA subscribes to the template repository where all the process templates are stored, which in turn notifies the UA about the availability of each new process template. Upon receiving the notification, the UA prepares a short description outlining the functionalities and potential charges of the process template and sends it to the user in the notification message (e.g., SMS). If the user is interested in the process template, the UA will contact the template repository to deliver an XML document of this process template, which is going to be stored in the user's mobile device upon reception for later configuration.

The client submits a configured process template (i.e., user's personalized composite service) to the UA, which is responsible for executing the service, collecting service results, and delivering results to the user. The UA accomplishes this by interacting with other agents in the architecture, which is described in the following.

Context Agent. The context agent (CA) is responsible for retrieving the context information and disseminating it to other agent in order to personalize the service creation and provision processes. As we have previously described, the CA maintains three kinds of contexts, namely user context, device context, and service context. The CA collects the context information from context providers and since there are three kinds of contexts there are also three kinds of context providers. Since it is

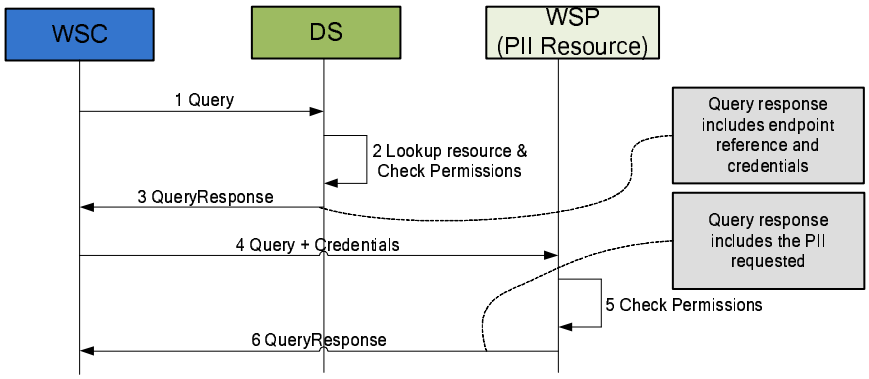


Fig. 2 Liberty Identity Web Services Framework

quite straightforward to provide device context (e.g., UAProf) and service context (e.g., service monitoring application), we will focus on user context providers.

User context providers are asked for PII regarding the users. Three different entities are possible in the role of user context provider: the telecom operator, third parties and the users themselves. Telecom operators provide dynamic user information related to the mobility such as the user location and the presence status. It can also provide more advanced data such as billing information or even charging information (micro-payments) which are closely related to the (financial) identity of the user. Following a similar model, third parties can participate in providing the CA with some specialized PII such as contact agenda or list of friends. Finally, end users can also provide static information such as their preferences at registration time.

The CA consists of a set of configurable context collectors. Each context collector handles one type of context information and encapsulates the details of interactions with the context provider for that information (e.g. the context collector pulls the context information periodically from the context provider).

Liberty Identity Web Services Framework (ID-WSF) [35] provides a means to communicate between context providers and context collectors. Moreover, ID-WSF architecture allows for a greater degree of abstraction supporting dynamic discovery and binding of context providers. ID-WSF basically defines three entities that participate in an identity transaction as shown in Figure 2: a Web service identity provider (WSP) that offers PII, a Web service identity consumer (WSC) that consumes the PII, and a Discovery Service (DS) that plays the role of a broker putting both in contact. Whenever a WSC needs a type of PII, it asks the DS about a WSP storing the requested information. The DS checks whether the WSC is allowed to access the information and if so, provides it with the address of the WSP and some credentials to access the information requested. Eventually, the WSC directly requests the WSP for the PII, showing the credentials delivered by the DS.

ID-WSF also provides some other advanced features such as an Interaction Service that is able to prompt the user for permission whenever it is needed: if the WSP is not sure about whether it should allow the WSC to retrieve the PII, it asks the user

to explicitly grant access. Therefore, it provides a high degree of protection when it comes to the management of users' PII.

Orchestration Agents. The orchestration agents include a set of agents, facilitating the asynchronous, distributed, and context-aware execution of composite services. These agents are *deployment agent*, *event agent*, and *service agent*. The deployment agent (DA) generates orchestration tuples, which will be described later, from the specification of personalized composite services that are submitted from the user agent (UA). The DA then deploys (uploads) these orchestration tuples into the *tuple spaces* of the corresponding services.

Service agents (SAs) act as proxies for Web services, monitoring and coordinating service executions. The knowledge required by an SA is a set of orchestration tuples, stored in a tuple space associated with the SA. The orchestration tuples are generated and deployed by the DA. There is one SA per service. For each Web service in the service layer, the service administrator needs to download and install SA, together with a tuple space.

The event agent (EA) is responsible for disseminating events. The EA maintains the information of events supported by the platform, i.e., for a specific event, what context attributes are relevant to this event and what condition should be satisfied to fire the event. For example, event `failed(`*s*`)` indicates that an execution failure of service *s* has occurred. The related context of this event is `executionStatus` and the condition, `executionStatus="failed"`, should be satisfied in order to fire the event.

Orchestration Tuples. The orchestration of a composite service is encoded in the form of *orchestration tuples*. Orchestration tuples are expressed as event-condition-action (`E[C]/A`) rules specifying the actions that must be performed when specific events occur and when the conditions hold. We introduce three types of orchestration tuples to coordinate the execution of personalized composite services:

- *Precondition tuples* specify the conditions that need to be satisfied before the execution of a service,
- *Postprocessing tuples* specify the actions that need to be performed after the execution of a service, and
- *Exception handling tuples* specify the instructions that dynamically react to runtime exceptions (e.g., mobile device disconnection and services failures).

For example, `unpresentable(r,d)[true]/transform(r,TS,d)` is an exception handling tuple that indicates that if the service result *r* cannot be displayed in the user's current device *d*, the result will be sent to *TS*, a transformation service, for adaptation.

The orchestration tuples of a composite service are statically derived by analyzing the information encoded in the statechart of the service (e.g., control flow and data dependencies, exception handling policies, and personal preferences).

Orchestration Interactions. Figure 3 (a) is a sequence diagram showing the process of the orchestration of personalized composite services. Firstly, the DA takes as input the specification of a personalized composite service from the UA, generates

orchestration tuples for each task of the personalized composite service, and injects these orchestration tuples into the tuple spaces of the corresponding SAs. Then, SAs parse the orchestration tuples and retrieve relevant information (e.g., events, conditions, and actions). The events (e.g., `lowBattery`) are registered to the EA, which in turn subscribes relevant conditions (e.g., `batteryRemaining < 15%`) to the context agent (CA). The EA fires and distributes events if the corresponding conditions are matched (e.g., when the current battery capacity of the device is less than 15% of its full battery power). Upon receiving the notifications (i.e., the occurrence of the events) from the EA, the SAs extract the corresponding orchestration tuples from the associated tuple spaces, evaluate the conditions, and perform the proper actions (e.g., service invocation in the diagram).

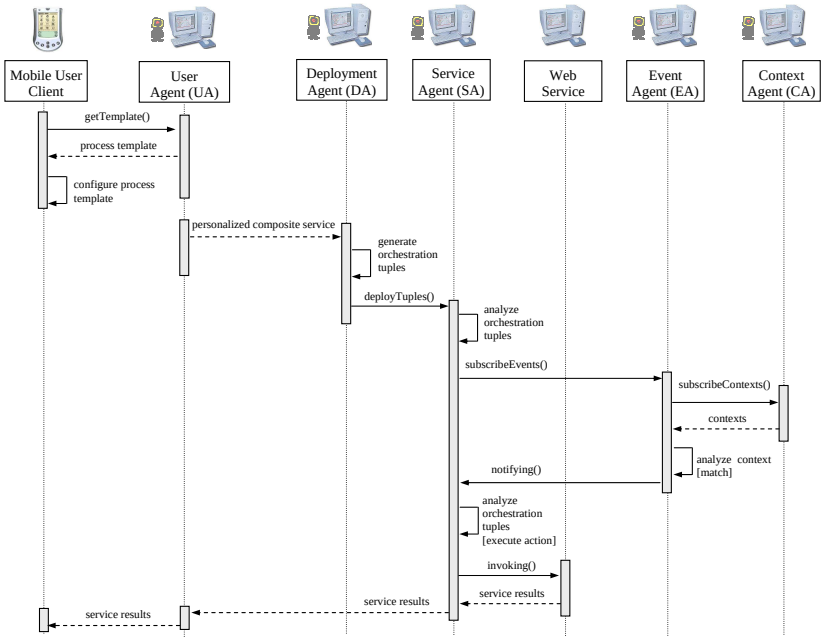
4.2 Service and Network Layer

The service layer hosts both the actual service implementation and the exposed Web service interfaces. On top of the convergent telecom IP core network, standardized middleware like IMS, Parlay/ParlayX and JSLEE provide advanced network control functionalities to bearer capabilities, call/session, and applications/services, and open up network resources by means of reusable telecom service enablers.

IMS can host three kinds of application services: the SIP application server, OSA application server, and the Camel service environment. These servers interact with the serving call session-control function (S-CSCF), home subscriber server (HSS), and multimedia resource function control (MRFC) with SIP protocol [23]. OSA/Parlay is a set of open APIs for easy access to core network capabilities from outside of the network. ParlayX evolves the OSA APIs into simplified Web service interfaces. As we have described in the previous paragraph, the OSA application server can also reside on top of the IMS control layer and becomes a part of the IMS architecture. JSLEE is the Java component and container specification for the event-driven, high-performance service logic execution environment (SLEE). It is worth noting that JSLEE can be used for both IMS SIP application servers and OSA application servers.

In the service interface sub-layer, ParlayX evolves the OSA APIs into simplified Web service interfaces, and OMA defines specifications of service enabling functionalities such as presence, instant messaging and location. From the Internet side, services are hosted and executed in different kinds of application servers such as J2EE, .NET and CORBA.

The network layer provides physical communication infrastructure. As we can see from Figure 1, hand-held devices of end users can physically access services through radio access networks (e.g., GPRS, MTS, and CDMA), or wireless Internet protocols (e.g., WiFi and WiMAX). It is clear that although we have services from both the telecom network and the Internet, at the service interface layer, they are converged under a unified Web service technology umbrella.



(a)

(b)

Fig. 3 (a) Sequence diagram of personal service orchestration (b) Screenshots of process template configuration

5 Implementation and Evaluation

To test the applicability of our architecture, we implemented a prototype system. We developed a process template builder, which assists template providers or mobile

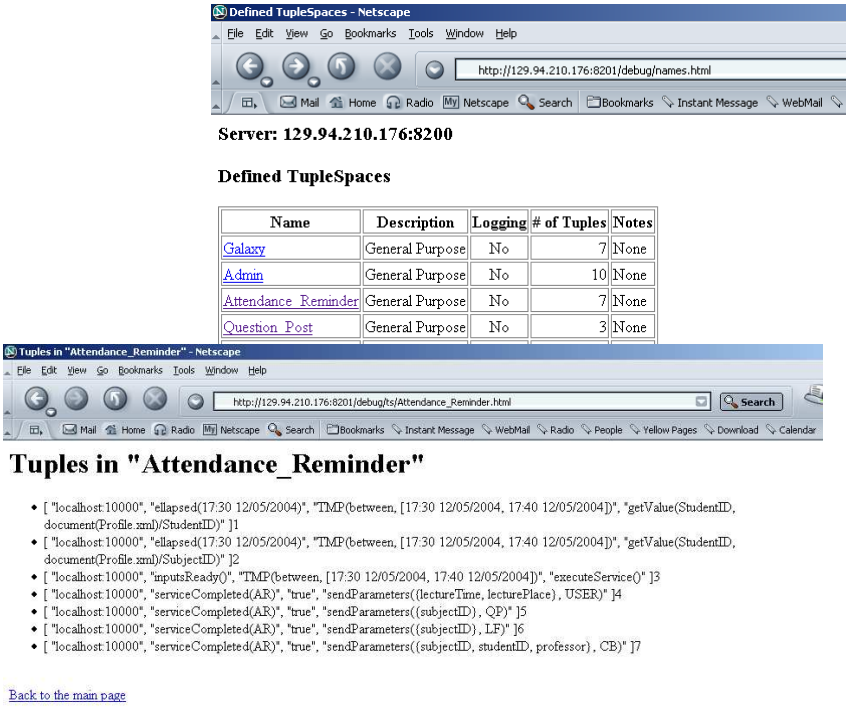


Fig. 4 Control tuples implemented using IBM TSpaces

users in defining and editing process templates. The template builder offers a visual editor (an extension of [33]) for describing statechart diagrams of templates. It also provides means to locate services in the registry. The client was implemented using J2ME Wireless Toolkit 2[12]. kXML 2[13] is used to parse XML documents on mobile devices and kSOAP 2.0[14] is used by the client to handle SOAP messages. Figure 3 (b) shows the screenshots of process template configuration.

Currently, the functionalities of agents are realized by a set of pre-built Java classes. In particular, the class *deployAgent* (for the deployment agent) provides method called `deploy()` that is responsible for generating orchestration tuples from composite services. The input is a personalized composite service described as an XML document, while the outputs are orchestration tuples formatted as XML documents as well. The orchestration tuples are then uploaded into the tuple spaces of the corresponding service agents. IBM's TSpaces[15] is used for the implementation of tuple spaces. IBM TSpaces is a network communication buffer

[12] `http://java.sun.com/products/sjwtoolkit`
[13] `http://kxml.enhydra.org`
[14] `http://ksoap.objectweb.org`
[15] `http://www.alphaworks.ibm.com/tech/tspaces`

Questions	Responses		
	A	B	C
Suppose you are invoking a remote service using a PDA and the invocation will take some time, which action you prefer to take: (A) wait with the handheld on till receive the result; (B) turn off the handheld and catch the results in another time	6	12	N/A
Suppose you are invoking a service using a PDA and the service needs some inputs, which strategy is your favorite to supply values for the service inputs: (A) manually input using stylus; (B) automatically collect the data (e.g., from user profile) (C) does not matter	1	15	2
Suppose you are receiving the results of a service using a PDA, you would like to receive: (A) all of them; (B) only important and necessary ones (C) does not matter	6	11	1

Table 1 Evaluation results of system design principles

with database capabilities (see Figure 4). Orchestration agents communicate asynchronously through the shared spaces by writing, reading, and taking control tuples. Each tuple is implemented as a vector of Java objects.

To validate our design of the system architecture, we conducted a usability study. We presented our system to 18 people, all from different educational backgrounds (8 undergraduate students, 2 masters students, and 8 PhD students) and computer literate. The presentation includes a powerpoint show of the architecture overview, a demonstration of the usage of the system, and `classAssistant`, the prototype application built on top of the architecture. The participants were then asked to use the system and given a questionnaire to report their experience in using the system.

Table 1 shows some of the questions from the questionnaire and the participants' responses. The responses actually validate and highlight some important design principles of the architecture: i) users should avoid data entry as much as possible, ii) the interactions during service execution should be asynchronous, and iii) the bandwidth consumption should be minimized. It should be noted that this usability study was conducted in a relatively small scale environment. Currently, we are planning a larger scale usability study of the system. More experimental results (e.g., performance study) are not reported here due to space reasons. Readers are referred to [32] for a detailed description of the system evaluation.

6 Related Work

Service personalization has recently become a very promising research area because it paves the way to a more widespread usage, creation, and customization of services. Its goal is to bring users from a passive role (i.e., content and service consumer) to an active role (i.e., content producer and service creator). Service creation technology is the key criterion which represents how user interacts with the system to create or compose a new service. Since ease-of-use is a key success factor of a service creation approach, we group different service creation approaches in increasing order of ease-of-use, namely: i) script-based, ii) rule-based, iii) choreography-based, iv) template-based, and v) natural-language based.

Service personalization performed by means of domain-specific scripting languages [26] is generally hard to use for developers. Rule-based approaches based

on ECA rules [25] or policy languages [24] have been used in telecom world for personalization of value-added services. While rules are easy to set, it is quite difficult for end users to foresee possible undesired side effects due to rules conflicts. Choreography-based approaches are gaining their momentum in the end-user service creation world, as proved by the increasing popularity of mashups and the birth of various environments for the intuitive non-expert creation of Web based information services, driven by the big companies, such as Yahoo! Pipes or Microsoft Popfly. These environments present graphical tools based on drag-and-drop interfaces which allow the user to create this little web-based applications even without any computing knowledge.

The natural language approach aims at deriving the formal specification for a service composition starting from an informal request expressed in natural language. Such formal, machine-understandable specification can then be used as input for providing the composite service through automated reasoning, like backward-chaining [8]. The core idea behind this approach resides in matching textual tokens within the natural language request to a semantic properties embedded in the service description, usually represented with labels annotated on the service interface: an implementation of this idea in the telecom domain was provided by the SPICE service creation environment [4], where natural-language goals are interpreted by an Automatic Composition Engine (ACE) in order to obtain a set of suitable compositions of semantic-annotated services [34]. Unfortunately, natural-language based techniques such as [14] are still limited to the predefined vocabularies and ontology concepts, scalability, and the intrinsic difficulties in handling the ambiguities of natural languages.

Our process templates approach aims at simplifying choreography approach, offering different common templates to be completed by user's data, thus providing a easier-to-use interface. Users can select a process template which can be seen as a pre-defined standard workflow (i.e. service composition). Related works like [36] proposes that choreographies and orchestrations (internal implementations) are described using a particular form of Petri Nets. Also the approach in [1] is considered as a choreography approach, where end-to-end web services composition methodology is realized by putting together semantically-annotated web service components in a BPEL flow.

In fact, very little work is being done on Web services orchestration for the benefit of mobile users. A middleware named Colomba [5] handles the dynamic binding for mobile applications. This middleware addresses multiple issues such as frequent disconnection, limited supply of battery power and absence of network coverage. A personal process-oriented system for mobile users is presented in [22], which investigates the specification and querying of processes that involve personal tasks and data. The objective of these works is to support simple applications for mobile users, rather than providing personalized specifications and adaptive orchestrations of composite services, fulfilling complex user requirements.

PerCollab [12] is a middleware system that integrates multiple communication devices with workflow systems. Relying on a centralized BPEL engine (which could lead to bottlenecks) and a context service, tasks can be proactively pushed to users.

However, PerCollab does not consider the personalized specification of business processes, nor distributed orchestration of the processes.

The emerging semantic Web efforts such as OWL-S[16] and WSMF (Web Service Modeling Framework) [18], promote the use of ontologies as a means for reconciling semantic heterogeneity between Web resources. Such efforts focus on designing rich and machine understandable representations of service properties, capabilities, and behavior, as well as reasoning mechanisms to select and aggregate services. The issues addressed in this area are complementary to those addressed in our work.

Finally, we also note that some industrial efforts in mobile Web services such as IBM's Web Service Toolkit for Mobile Devices[17] and Microsoft and Vodafone's Mobile Web Service initiative[18]. The former provides tools for developing Web services on mobile devices while the latter plans to create new standards to integrate IT and mobile worlds through Web services.

7 Conclusion

While much of the work on Web services has focused on low-level standards for publishing, discovering, and provisioning Web services in wired environments and for the benefit of stationary users, we deemed appropriate to put forward novel solutions and alternatives for the benefit of mobile users. In this chapter, we have presented the design and implementation of a layered, multi-agent framework that enables personalized composition and adaptive provisioning of Web services. Our system builds upon the building blocks of Web services, agents, and publish/subscribe systems and provides a platform through which services can be offered to mobile users. One possible extension to our current system is a mechanism for seamlessly accessing services among multiple computing devices. Indeed, during the invocation of a Web service, especially one having long running business activities or with complex tasks (e.g., composite services), users are more likely to be switching from device to device (e.g., from office PC to PDA). Applications can not be allowed to terminate and start again simply because users change devices and users should not experience a break during the service invocation while they are moving. This is extremely important for the people in time critical working environments (e.g., doctors in hospitals). A second possible extension is to support team work so that multiple (mobile) users can virtually collaborate with each other in the same business processes. Finally, we plan to extend the architecture to support large-scale environments, and building more mobile applications on top of the architecture to further study its performance.

[16] http://www.daml.org/services/owl-s

[17] http://www.alphaworks.ibm.com/tech/wstkmd.

[18] http://www.microsoft.com/serviceproviders/mobilewebservices/mws_tech_roadmap.asp.

References

1. Vikas Agarwal, Koustuv Dasgupta, Neeran Karnik, Arun Kumar, Ashish Kundu, Sumit Mittal, and Biplav Srivastava. A Service Creation Environment based on End to End Composition of Web Services. In *Proc. of the 14th International Conference on World Wide Web (WWW'05)*, pages 128–137, Chiba, Japan, May 2005.
2. Sudhir Ahuja, Nicholas Carriero, and David Gelernter. Linda and Friends. *IEEE Computer*, 19(8):26–34, August 1986.
3. Tony Andrews et.al. Business Process Execution Language for Web Services 1.1. `http://www-106.ibm.com/developerworks/library/ws-bpel`.
4. Mariano Belaunde and Paolo Falcarin. Realizing an MDA and SOA Marriage for the Development of Mobile Services. In *Proc. of the 4th European Conference on Model Driven Architecture (ECMDA08)*, Berlin, Germany, June 2008.
5. Paolo Bellavista, Antonio Corradi, Rebecca Montanari, and Cesare Stefanelli. Dynamic Binding in Mobile Applications: A Middleware Approach. *IEEE Internet Computing*, 7(2):34–42, March/April 2003.
6. Boualem Benatallah, Quan Z. Sheng, and Marlon Dumas. The Self-Serv Environment for Web Services Composition. *IEEE Internet Computing*, 7(1):40–48, January/February 2003.
7. Djamal Benslimane, Schahram Dustdar, and Amit P. Sheth. Service Mashups. *IEEE Internet Computing*, 12(5), September/October 2008, to appear.
8. Alessio Bosca, Giuseppe Valetto, Roberta Maglione, and Fulvio Corno. Specifying Web Service Compositions on the Basis of Natural Language Requests. In *Proc. of the 3rd International Conference on Service Oriented Computing*, Amsterdam, The Netherlands, December 2005.
9. Gonzalo Camarillo and Miguel-Angel García-Martín. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*. 2nd Edition, John Wiley & Sons Ltd., 2006.
10. L.J. Camp. Digital Identity. *IEEE Technology and Society Magazine*, 23(3):34–41, 2004.
11. M. Caporuscio, A. Carzaniga, and A. L. Wolf. Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications. *IEEE Transactions on Software Engineering*, 29(12):1059–1071, December 2003.
12. Dipanjan Chakraborty and Hui Lei. Extending the Reach of Business Processes. *IEEE Computer*, 37(4):78–80, April 2004.
13. James Clark and Steve DeRose. XML Path Language (XPATH) Version 1.0. http://www.w3.org/TR/xpath, November 1999.
14. Kurt Englmeier, Javier Pereira, and Josiane Mothe. Choreography of Web Services Based on Natural Language Storybooks. In *Proc. of the 8th International Conference on Electronic Commerce (ICEC'06)*, Fredericton, New Brunswick, Canada, August 2006.
15. Graham Klyne et al. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. `http://www.w3.org/TR/CCPP-struct-vocab`, visited on 24 June 2008.
16. Hal Lockhart et al. Web Services Federation Language (WS-Federation). `http://www.ibm.com/developerworks/library/specification/ws-fed/`, visited on 24 June 2008.
17. James J. Garraham et al. Intelligent Network Overview. *IEEE Communications Magazine*, 31(3):30–36, March 1993.
18. Dieter Fensel and Christoph Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.
19. Marit Hansen, Ari Schwartz, and Alissa Cooper. Privacy and Identity Management. *IEEE Security and Privacy*, 6(2):38–45, 2008.
20. David Harel and Amnon Naamad. The STATEMATE Semantics of Statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, October 1996.
21. Richard Hull, Bharat Kumar, and Daniel Lieuwen. Towards Federated Policy Management. In *Proc. of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, Lake Como, Italy, June 2003.

22. San-Yih Hwang and Ya-Fan Chen. Personal Workflows: Modeling and Management. In *Proc. of the 4th International Conference on Mobile Data Management (MDM'03)*, Melbourne, Australia, January 2003.
23. Hechmi Khlifi and Jean-Charles Grégoire. IMS Application Servers: Roles, Requirements, and Implementation Technologies. *IEEE Internet Computing*, 12(3):40–51, May/June 2008.
24. Patricia Lago. A Policy-Based Approach to Personalization of Communication over Converged Networks. In *Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, Monterey, California, USA, June 2002.
25. Carlo A. Licciardi, Gianni Canal, Alessandra Andreetto, and Patricia Lago. An Architecture for IN-Internet Hybrid Services. *Computer Networks*, 35(5):537–549, April 2001.
26. Carlo A. Licciardi and Paolo Falcarin. Analysis of NGN Service Creation Technologies. *IEC Annual Review of Communications*, 56:537–551, November 2003.
27. Swee B. Lim and David Ferry. JAIN SLEE 1.0 Specification. `http://jcp.org/en/jsr/detail?id=22`, visited on 20 June 2008.
28. Rebecca Montanari, Emil Lupu, and Cesare Stefanelli. Policy-Based Dynamic Reconfiguration of Mobile-Code Applications. *IEEE Computer*, 37(7):73–80, 2004.
29. Tim O'Reilly. What is Web 2.0. `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`, visited on 27 June 2008.
30. Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer*, 40(11):38–45, 2007.
31. Mike P. Papazoglou and Willem-Jan van den Heuvel. Service Oriented Architectures: Approaches, Technologies and Research Issues. *The VLDB Journal*, 16(3):389–415, 2007.
32. Quan Z. Sheng. *Composite Web Services Provisioning in Dynamic Environments*. PhD thesis, The University of New South Wales, Sydney, NSW, Australia, 2006.
33. Quan Z. Sheng, Boualem Benatallah, Marlon Dumas, and Eileen Mak. SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment. In *Proc. of the 28th International Conference on Very Large Databases (VLDB'02)*, Hong Kong, China, August 2002.
34. Mazen Shiaa, Paolo Falcarin, Alain Pastor, Freddy Lécué, Eduardo Silva, and Luis F. Pires. Towards the Automation of the Service Composition Process: Case Study and Prototype Implementations. In *Proc. of the ICT Mobile and Wireless Communications Summit*, Stockholm, Sweden, June 2008.
35. Jonathan Tourzan and Yuzo Koga, Ed. Liberty ID-WSF Web Services Framework Overview. `http://www.projectliberty.org/liberty/content/download/1307/8286/file/liberty-idwsf-overview-v1.1.pdf`, visited on 24 June 2008.
36. Wil M. P. van der Aalst and Mathias Weske. The P2P Approach to Interorganizational Workflows. In *Proc. of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, Interlaken, Switzerland, June 2001.
37. Phil Windley. *Digital Identity*. O'Reilly Media Inc., 2005.
38. Wireless Application Forum. Wireless Application Protocol User Agent Profile Specification. `http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf`, visited on 27 June 2008.
39. Juan C. Yelmo, José M. Del Álamo, and Rubén Trapero. Privacy and Data Protection in a User-Centric Business Model for Telecommunications Services. In *Proc. of the IFIP International Federation for Information Processing: The Future of Identity in the Information Society*, Karlstad University, Sweden, June 2008.
40. Qi Yu, Athman Bouguettaya, and Brahim Medjahed. Deploying and Managing Web Services: Issues, Solutions, and Directions. *The VLDB Journal*, 17(3):537–572, 2008.
41. Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, July 2003.
42. Johan Zuidweg. *Implementing Value-Added Telecom Services*. Artech House Inc., 2006.