6

The Way Forward – Paths to Follow

Chapter 5 has described how an IMS-enabled network can be exploited beyond its original conception to retain competitiveness and boost innovation in an operator business. That has been shown through the case of WIMS 2.0, which promotes the convergence of telecom services with the Web 2.0, based on a fully enabled IMS service architecture. This kind of strategy preserves the positioning of the telcos in the 2.0 era. However, the wide ICT landscape is a rapidly changing environment. This dynamic context requires flexible adaptation and an enthusiastic and firm pace to walk towards the future. New business models and new technologies modulate this evolution, without compromising cost optimization and exploitation of past investments, especially those in network infrastructure. This chapter describes and discusses potential paths to follow for operators looking to a future that is securing investments and profitability. In these paths of evolution, IMS-enabled service architecture plays a decisive role in supporting innovation and providing differentiation in service creation. After all, services and service construction are the essence in driving the digital evolution.

The present chapter presents service platforms and service creation trends in the mid term, as well as a set of initiatives in the R&D domain looking further into the future. The chapter takes the case of service platforms for operators first. Here, SDP, SOA, the marketplace of services and the open telco initiatives are discussed. Then the service creation is considered as this is an intricate aspect of the success of services and products. Service creation is examined from the perspective of all key contributors, including the user participating in the service generation, the professional developers and the challenges posed by devices in service development.

IMS: A Development and Deployment Perspective Khalid Al-Begain, Chitra Balakrishna, Luis Angel Galindo and David Moro

6.1 Operators and Service Platforms

Innovation is an important asset for operators to exploit their service capabilities, which might not be explicitly exposed to service development and product building. There is a hidden potential in telecom networks for service creation that today is not fully exploited. In most industrialised countries, market saturation is leading to new technology solutions that target at leveraging already existing capabilities for new usages. In the technology sphere, this is materialized by a new service platform that unveils the potential of existing telecom architectures and underlying networks for service creation. As such, IMS hides a potential for service innovation that can be opened to the large ecosystem of service creators – both in-house and third parties – with the right tools and the right service platforms. If not, investments in IP multimedia will be relegated to pure technology trials.

This section reviews the trends for opening telco service capabilities to a wider space of third party developers in the so-called 'open telco' initiatives that are based on service platforms. It also discusses how these initiatives can be supported by service delivery platforms (SDP), which are geared beyond the open capabilities objective to also target reducing time-to-market for in-house application development. Finally, there is the creation or the telco contribution to a digital marketplace of open services and capabilities. Throughout the section, the role of IMS in opening service capabilities and flexible commercialization of open services in a marketplace has been sustained. It is argued that it provides new service capabilities and enablers to the catalogue for the sake of operator competitiveness and positioning in a fierce market of open services and service creation.

6.1.1 Open Capabilities

Operators are in a difficult situation in the digital service landscape. Their business is exposed to a continuous risk of saturation, and growth principles are not reflected on the seeds of seducing customers with appealing services. The majority of the income in the residential markets comes from basic services like traditional telephony and increasing bandwidth for pure Internet connectivity. Even though operators are in the loop of continuous developing and bringing a myriad of new value-added services, customers are still using the same basic services like messaging and basic voice calls, thus ignoring the remaining 90% of the service portfolio. Instead of these value-added services, customers seem to use the Internet services. Therefore, it can be concluded that operators are not engaging correctly with the user population in the value-added services and applications. Possibly due to that their business models are not attractive enough, and that they are disconnected from user needs and desires in terms of digital lifestyle. Services offered by telcos are not as innovative as those populating the wide Internet, where different patterns for building services, open innovation, cheaper development, proximity to users, blurry frontiers between users and developers are some of the bases of this effect.

There is an emerging ecosystem on application stores, software-as-a-service and a cloud full of service capabilities offered to Internet communities for development. This indeed channels and stimulates the creative energy of digital users. Some remarkable players of this trend in opening services and capabilities are the key Internet players, such as Amazon, Google, Yahoo and Salesforce. Each of them is targeting different segments: from platform

and storage like Amazon, or CRM and productivity applications like Salesforce, to web Internet services like mail, maps, search which Google and Yahoo are offering.

On the other hand, telco companies have unique and valuable assets that potentially could be mashed up with the others in this cloud of service capabilities. This implies a change of thinking in the telecom business in the form up to the Internet trend. This means that the telcos would offer a portfolio of open capabilities in the form of reusable APIs, tools and SDKs to the wide Internet, to let developers and software companies of different sizes integrate telco services in new and innovative applications. An operator pursuing this new business model and providing the necessary platform for service creation and mash-up with features coming from other service providers can consequently qualify to become an 'open telco'.

By delivering this kind of platform, the 'open telco' obtains an important source of benefits for the local brands; customers and enhancements on time-to-market; and CAPEX utilization and income. A twofold principle for generating income governs this change of business thinking:

- An 'open telco' is able to create a new segment of customers, the developers, willing to pay for – or commit to – a set of high level assets that enable them to build new services.
- An 'open telco' can create new revenue streams based on the customer usage of new third party applications built on its open capabilities. Hence, customers will enjoy a new wave of services, increasing traffic and access business, which is shared with the third party that delivers and maintains each application.

The success of such 'open telco' initiatives depends on a set of factors that are indeed not technological. These include:

- Creating a lively community of developers to produce applications with a fair revenue-share model with the 'open telco'. Beta testers of the developed applications should also be engaged and encouraged. Thus, the community is the axial part of such an 'open telco' initiative, at least to foster the initial open platform utilization and to exploit marketing proximity between users and developers.
- Application store and customer access to innovation is an important part. Beta testers are the first front end for a new application trial, but the necessity to step into commercialization of a new application is the key for generating real revenue and traffic in the operator systems. Those applications with great impact in the market are expected to be included in the operator's portfolio and branding of the 'open telco' upon achieving a fair agreement with the owner. Thus, if an 'open telco' initiative achieves a critical mass, it can determine the agenda for future products of the associated telecom company, and probably others by propagation.

Nevertheless, there are drawbacks in the 'open telco' strategy, mainly derived from the clash of business models between the source, the telco company, which is naturally a closed and controlled business, and the Internet ambience, which is the initiator of the 'open' service provider trend. Since the Internet players are offering similar open platforms that usually do not look for direct revenue from users and developers and their services are not subject to a tariff, the success of an 'open' but 'priced' initiative such as those promoted by 'open telco' can reduce their acceptance possibilities to a minimum. Another remarkable factor

for the potential failure of the 'open telco' initiative as a global trend is fragmentation. Telcos operate in geographically fragmented markets, while the Internet is regarded as virtually global. When examining the market situation, there is an increasing panorama of telco companies joining the 'open' trend. Within the established and de facto principles of an 'open telco' described here, there are variations from company to company when examining how they approach the community. The number and functionality of the offered open APIs, the technology and the business models behind are the substantial differences. In a comparative summary of a survey that has been conducted for a number of telcos, the most representative were: Telefonica's Open movilforum, O2 Litmus, Vodafone Betavine, Orange Partner, BT's Ribbit and Telenor Playground.

Table 6.1 summarizes the differences in the business models that were found in the above mentioned open initiatives. In particular, the table shows if the open community of a given telco recognizes the role of the developer and the role of beta tester, and if there is a channel to commercialize the developments via an application store, and how the parties are charged or revenue is shared.

From a pure technological viewpoint, there are commonalties and best practices to be recommended. The Web 2.0, as the first step towards a future Internet of services, has already shown that the Internet is 'The Platform' for constructing and delivering ubiquitous services through the dynamic mash-up of open services coming from different players, which exposes functionality of their services by means of open APIs. Functionality exposed by the 'open telco' is very dependent on its strategy and technical plans. However, there are commonalties between the different open telco initiatives in the sense that most of them are exploiting the base telco service assets like messaging, location, phone book and telephony features. Table 6.2 shows a comparison of the open capabilities offered by the studied open telco initiatives.

Table 6.1 Open business model comparison.

Initiative	Developer	Beta tester	Application store
Vodafone Betavine	Free of costs	Free	Yes, not linked to Betavine
Orange Partner	Price is dependent on the specific API; alpha and beta APIs are free	Role not defined	Orange Shop; revenue shared with developer
Telenor Playground	Free of costs	Not defined	Yes; revenue shared upon negotiation
Ribbit	Prices dependent on the planned usage	Not defined	Widget platform planned; revenue shared
Open movilforum	Charged per use to an MSISDN; free vouchers	Not defined	No
O2 Litmus	Free of cost	Yes	Yes; revenue share 70% for the developer

Data collected in January 2009.

Table 6.2 Open capabilities comparison.

	Vodafone Betavine	Orange Partner	Telenor Playground	BT Ribbit	Open movilforum	O2 Litmus
SMS send	Yes	Yes	Yes	Yes	Yes	Yes
SMS reception	No	Yes	Yes	Yes	Yes	No
MMS send	Planned	No	Yes	No	Yes	No
MMS reception	No	No	Yes	No	No	No
Call control	No	Yes	Yes	Yes	No	No
Multimedia conference	No	Yes	No	Yes	Yes	No
Device capabilities	Yes	Yes	No	No	No	No
Location	No	Yes	Yes	No	Yes	Yes
Wap push	No	No	No	No	Yes	No
Presence	No	No	No	No	Yes	Yes
Phone book	No	Yes	No	Yes	Yes	No
Voicemail	No	Yes	No	Yes	No	No
Authentication	No	Yes	No	Yes	No	Yes

Data collected in January 2009.

To retain relevance and leverage of its unique service assets, an 'open telco' operator offers its capabilities towards this new third ecosystem, thus enabling the easy introduction of telco functionalities within almost any third party application. The latter is not limited to just web applications but also to other non-web applications that simply use HTTP as a transport mechanism to join the Internet service fabric. The open APIs exposing the telecom capabilities should also be offered to third parties and to the Internet in a controlled and adapted manner. The open API concept can be used to meet both in-house and external service delivery needs. To design a future-proof solution, an open API model of telecom services has to have a web or HTTP-like view on them. To define open APIs of telecom capabilities, the combination of the representational state transfer (REST) (Fielding, 2000) philosophy (defined in Section 6.2.1) with blended operations of an RPC (remote procedure calls) is recommended. This will provide a simple-to-use API while still having a fine control of session-oriented services like telephony, which requires a richer set of operations to be exposed. Besides the remote API, a common practice in this open environment would be to provide a set of client libraries in different programming languages such as Java, PHP, Python, Ruby, etc. These libraries will handle the HTTP specifics of the invocation of API primitives from the application side (client side) towards the remote API server in the open platform of the operator. This can accelerate development and integration of the API into the application, becoming in fact a matter of attracting developers into the community of the given 'open telco'.

Table 6.2 describes the different approaches taken by the studied 'open telcos' for using remote API technology and the languages for the accompanying client libraries.

Besides the pure exposure of telecom capabilities by means of open APIs, the platform offered within an open telco initiative can be enriched with SDKs and other tools for easing mash-up generation for nonprofessionals users, i.e. user-generated service (UGS) tools. The open platform should also rely on mechanisms for registering and promoting

Table 6.3 API technologies and client libraries in open initiatives.

Initiative	API technology	Client library languages
Vodafone Betavine	REST	Python, Java, Ruby
Orange Partner	RPC (for some APIs); REST-like (for some APIs)	Java .NET
Telenor Playground	RPC: Parlay X (SOAP)	Java
Ribbit	REST	Flex (ActionScript 3) and Flash
Open movilforum	REST-like	PHP, Ruby, Python
O2 Litmus	RPC: SOAP	Java, .NET, Flex and PHP

Data collected in January 2009.

applications, i.e. the application store, and tools for managing the marketplace of services and applications behind the community. In this context, REST and other development techniques, UGS and digital marketplace tools are also discussed in this chapter The fact that IMS is an innovative technology and concept affects this kind of programme, providing a richer multimedia set of service features that can be exposed to augment the catalogue of open capabilities that operators are offering to the community of developers. Beyond that, the open communities are usually populated by pioneers in technology adoption, the so-called early adopters. New capabilities coming from the IMS world can be offered to a community eager to experience high end technology. This can be a twofold objective: on the one side, to maintain the sophisticated style of the community, as well as to gather a feedback outside the operator on service design patterns, usage concepts and other metrics about IMS services.

6.1.2 SDP and SOA

A service delivery platform (SDP) usually refers to a set of components within a telecom operator service architecture that is in charge of easing service creation, deployment and execution utilizing service capabilities and enablers. As the keystone objective of an SDP would be to reduce time-to-market of a service, the key characteristic for an SDP is thus the ability to ease the integration task of application logic with the enablers and capabilities in the telecom service architecture for the delivery of a service to users. The SDP thus enables the rapid transformation of a service concept into a full product including all the related service lifecycle management and commercial interactions. The SDP is positioned within the operator's service architecture, as described by the model in Figure 6.1, constituting the axial component of the service integration layer.

There is no global standard for SDP implementation, although there are some industrial de facto conventions. Upon the latter, a set of principles can be extracted for describing SDP following a compass-inspired interpretation. This is depicted in Figure 6.2.

- *North.* An SDP provides clear service interfaces for exposing telecom capabilities, both for in-house development as well as enabling external agents to develop services and applications using the operator as the platform, i.e. the service delivery platform. An example is the previously described 'open telcos' initiative, whose technical realization tends to be realized with the operator's SDP.

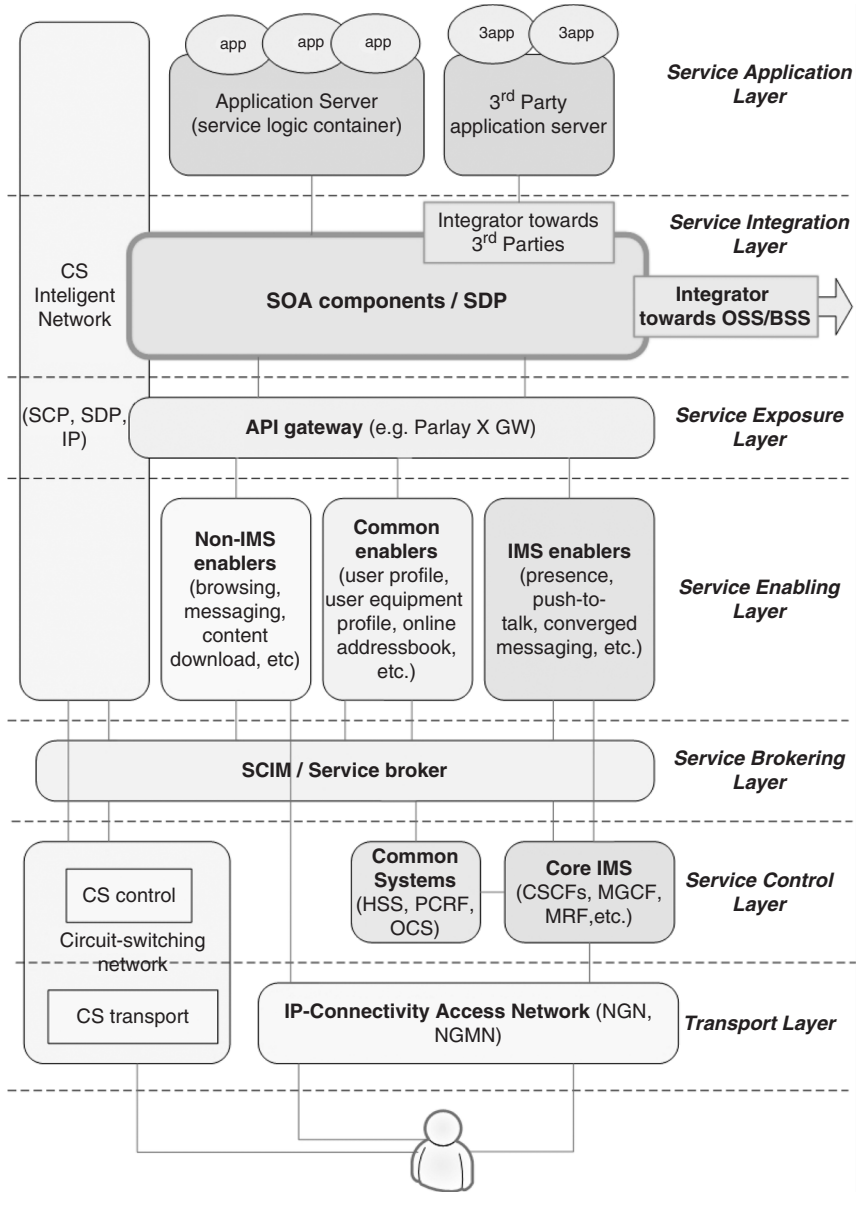

Figure 6.1 The SDP in the service delivery domain layered architecture.

- *South.* The SDP provides abstractions for the operator's enablers and service capabilities in the underlying service architecture, thus providing a simple means of invoking telephony, media and other service features from the applications by means of these abstractions. Therefore, the SDP simplifies the process of integrating an application container with the network infrastructure and enablers.

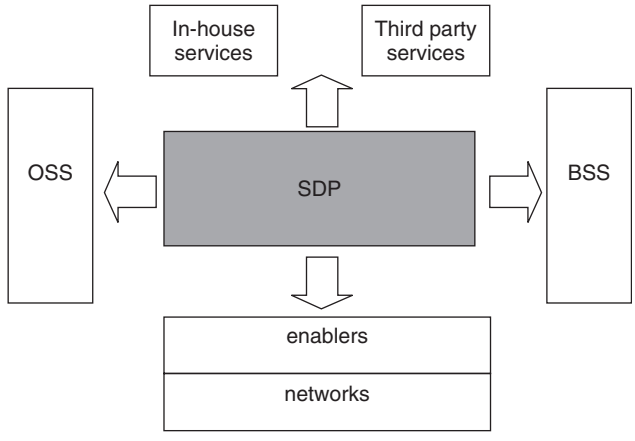

Figure 6.2 The compass-inspired approach for the SDP.

- *West and east.* The SDP includes connectors to the OSS and BSS systems of the operator, as well as for other back-office applications. By means of this connection to OSS/BSS, the services deployed over the SDP can be integrated into the company service and product lifecycle management, including operations that usually involve the service delivery architecture and the OSS and BSS systems, like the provision and activation of services; supervision and failure of management; administrative validations; billing and quality of experience (QoE) measurements. In this way, an SDP simplifies the integration of a new application towards the commercial and operation systems of the company.
- *Central.* The SDP typically provides a service creation environment for service design, and a service orchestration and execution environment, which represents the core of the SDP. Together with these core functions, there is a set of components in the SDP to enable policy control, security, user directories and other functions for business/operation process adherence and integration.

The adoption of the service-oriented architecture (SOA) (OASIS, 2008) in the SDP simplifies enabler deployment and facilitates other SDP requirements for composition, reuse, delegation, orchestration, policy enforcement and business/operation process adherence and integration. SOA has been traditionally associated with the IT domain, but now the trend is to reuse it in telco-oriented environments like the SDP. SDP and the election of SOA for its implementation have been deemed necessary in order to cope with a series of business drivers; in particular, the SDP focuses on the reduction of time-to-market for new services, as already mentioned. It is also expected that the SOA-based SDP allows the creation of a wider range of value-added services in a cost-efficient way, the implementation of an open service marketplace for third party interactions and the efficient integration with OSS/BSS.

The service-oriented architectures seek to save costs in investments and to provide flexibility to services offered by companies. SOA pursues a methodology to define and reuse software components and business processes that on many occasions are already

developed and implemented. Thus, SOA is a paradigm of software architecture with a service orientation as the prime design principle. SOA can be defined as an architecture that provides uniform methods to offer, discover, interact and govern capabilities, which are modelled as SOA services. These services are loosely coupled and present clear interfaces to match business requirements and implement reutilization. Due to its neutrality, in an SOA environment services are accessible irrelevantly of the type of platform and technology of implementation. Thus, SOA is not linked to a specific technology for its realization and there are many to satisfy the implementation requirements, e.g. SOAP, RPC, DCOM, CORBA, WCF. Although SOA is technology independent, web services are the most extended technology together with their associated technologies like HTTP, XML, WSDL, SOAP and UDDI. However, a service-oriented SDP usually provides support for a higher range of technology in order to guarantee integration with a large variety of services and capabilities. Beyond that, there are ongoing trends to enhance SDP to support service orientation as well as resource orientation, a software philosophy embodied in some interface techniques like REST.

For constructing the infrastructure of the SOA-based SDP, a skeleton of the platform can be based on the a SOA suite. Typically, these elements, although not standardized, are part of the SOA product portfolio of the IT vendors with a clear leadership in this domain. The common key elements in a SOA suite across vendors' portfolios are the BPM, ESB, Registry and Compositor. These elements are depicted in Figure 6.3 and described next:

- BPM (business process management) is a set of technologies that host and execute business processes. BPM workflows can be long-lived sessions and can involve human interactions.
- ESB (enterprise service bus) is a logical service bus to ease service access and interoperability by means of decoupling between applications, service enablers and capabilities, on the one side, and the integration of different systems, on the other side. ESB basic functionalities are message exchange and routeing, and translation between different technologies.

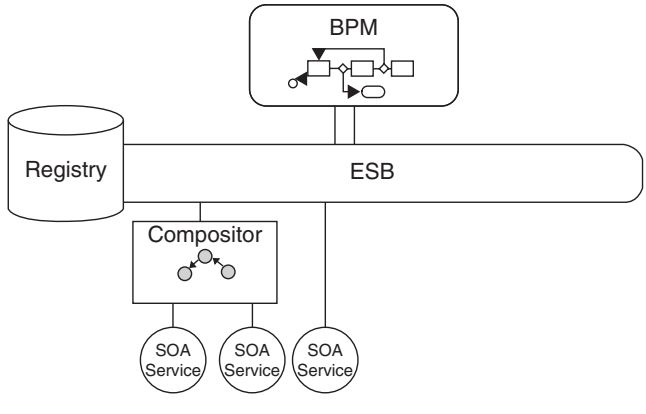

Figure 6.3 The elements of an SOA suite.

- Service registry provides the ability to register, discover and govern services in the service-oriented architectures. In run-time, the service registry is accessed by applications to locate and bind services. Thus, this registry contains a service catalogue and those parameters required for the execution of services.
- Compositor (or service orchestrator) utilizes and orchestrates different elements in the service architecture to produce combined services that are more complex and reusable by application. Sometimes, this is also referred as 'integrator', a piece of the SOA dedicated to the development of high transactional services based on reutilization of other services.

On top of these SOA elements, the functional components can be constructed, interlinked and connected to external pieces of the service delivery infrastructure and systems beyond the service delivery domain. Of all the elements, the ESB is regarded as the key element for providing the infrastructural support of the SDP, due to its integration ability.

A clear conclusion is that SDPs, which are usually implemented with IT technologies, will abstract the service enablers and service capabilities of a telecom network. Each of these technology domains is driven by different principles, where the telecom technological domain is usually based on asynchronicity and event-driven protocols, while the IT world, where SOA is positioned, is based on information exchange patterns and synchronicity. Usually the former is associated with optimized technologies like those associated with circuit switching, while the latter is rather oriented towards Internet-like processes.

The role of IMS in this clash of technology philosophies is precisely the ease of merging or abstracting capabilities from the telecom technology domain into the IT domain. This means that IMS capabilities should be easier to be abstracted and integrated into the SDP to become ready to be utilized by services developed on the SDP, since the technology foundations are the same or, at least, present a high degree of synergy. Besides, IMS provides a rich set of enablers and service capabilities that considerably augments the resulting portfolio of service features offered by the SDP to the service creators when building any new application over the SDP.

6.1.3 Digital Marketplace of Services

Nowadays, service providers recognize that access-line loss is progressing rapidly and needs the ability to introduce new services quickly in response to market needs generating new revenue streams. It is therefore clear that service providers can realize more revenue when SDP and business and operational systems (B/OSS) are fully integrated. With the back-office functions of provisioning, activation, service quality monitoring, customer intelligence and billing, service providers can deliver great customer experience and actively manage the service environment. Communications providers have an opportunity to satisfy consumer demands better through the new concept of a digital marketplace strategy, resulting in higher acquisition, retention and satisfaction rates. It begins by providing the digital environment through expertise and resources and delivering the digital lifestyle through applications, content, commerce and social networking. Creating a marketplace for next-generation services where Internet, media and telecommunications industries converge means that it requires the participation of service providers that have multiple specialities. These providers can make available distinct but complementary resources that can be combined and commercialized for a variety of customers using a variety of business models.

A digital marketplace is an advanced collaboration framework with three main pillars:

- The exposition of its capabilities from the network and OSS/BSS to third parties as reusable, manageable and chargeable services exposed through open APIs.
- The composition and aggregation of the exposed services with services offered by third parties.
- The agile marketing of these resulting products under several brand and business models beyond advertising, e.g. SLA-based business models. Taking these aspects into account, the digital marketplace of services can be defined as a business and management layer to be put on top of these 'service-oriented communities', where the participants expose, share and combine services to create new ones. The marketplace enables the end-to-end management of services from conception to cash across service provider's domains, the automated establishment of business agreements between the participants and the commercialization of these collaborative services and products under different brand and business models. Figure 6.4 shows the marketplace vision from Win Win Consultores (Consultores, 2009).

Since the collaboration and interworking between different industries is a key marketplace characteristic, standardization is essential: for instance, standardizing how a service provider can share its assets as services reusable by other service providers and, in the same way, can reuse the assets provided as services by other services providers without losing end-to-end control and management capabilities. In this manner, the marketplace concept is closely related to standards as the TM Forum (TMF, 2009b) service delivery framework (SDF) (TMF, 2009a). The aim of SDF is to enable service providers to be agile and competitive when managing the full service lifecycle in this new world, where services can either be developed by third parties or by themselves, and can also be customized, reprogrammed, composed, assembled with other services, etc. Therefore, the service delivery framework helps organizations maintain control over service lifecycle management across all execution environments (i.e. mobile, convergent telco, IPTV, broadband provider, etc.) and frees them to architect their SDPs as they choose. The operation of a marketplace can be explained by the following example:

- A digital marketplace exits where telcos and ISPs publish their services in order to allow other service providers to reuse these services. The service specifications include

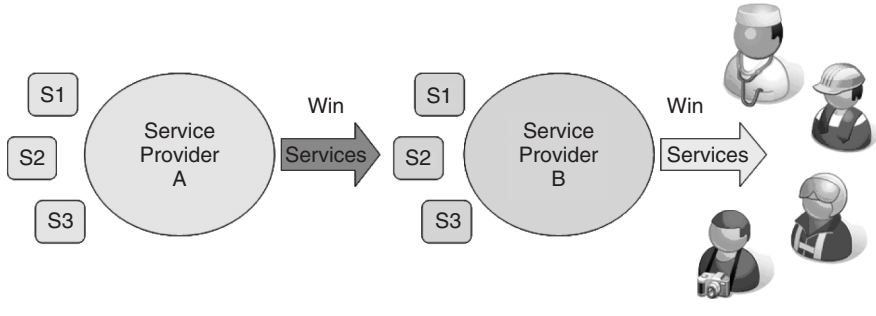

Figure 6.4 Marketplace by Win Win Consultores.

functional, management and business aspects like the price model, the payment model, QoS metrics, service levels, etc.

- An operator discovers in a digital marketplace the virtual meetings service from an ISP and decides to combine it with its high quality video on-demand service to create a new commercial service. As the first step, the operator and the ISP agree a service level agreement (SLA), including aspects like quality of service, support and troubleshooting model, brand model, business model and revenue sharing. Once the agreement is signed, the telco operator develops the new service and defines the related commercial offer for this service, which is published in the marketplace product catalogue.
- A customer discovers the product, contracts and uses it. The marketplace is in charge of monitoring the service level and the service usage (detecting SLA violations, sending events to the billing and service assurance systems, etc.), even when the services included in the product are distributed across the telco and ISP service provision domains. This new service is a win–win for both, the telco that offers a more attractive service and the ISP that reaches new markets and final clients who have access to an enhanced service.

As mentioned earlier, a marketplace enables the end-to-end lifecycle management of services (from conception to cash) across different service provider's domains, the automated establishment of business agreements between the participants and the commercialization of these collaborative services and products under different brand and business models. Therefore, platforms for opening the operator capabilities (like those behind Open movilforum, Vodafone Betavine, etc.) or tools for composing services (like Popfly, Pipes, etc.) are not a marketplace. There are several benefits provided by a digital marketplace of services:

- The marketplace is an enabler for collaborative innovation with other service providers and industries. The marketplace allows taking advantage of the collective intelligence in order to build more attractive services that use components provided by third parties.
- Time-to-market reduction. The marketplace owner can use its own services and services shared by others to create a new one.
- The digital marketplace is a new business with the possibility of reaching new market niches (the long tail).
- With the digital marketplace, the shop runner can have an enriched product portfolio for customers, enriched with services syndicated by other service providers. Moreover, customers do not have to worry about who is providing each individual service.
- Telco operators, who are expected to be the digital marketplace owners, will increase their revenues because other service providers will reuse their services to create new ones. New incomes are generated because of the increase in the network traffic and because of revenue-sharing agreements between service providers.

In addition to this, the marketplace adds value to all the participants in the environment, because it provides common functions like billing or service assurance, so the involved service providers do not have to take care to provide these functions to the final customers.

As service providers transition to an IP-based infrastructure, deploying an IP multimedia subsystem (IMS) network is a natural progression for telecom operators. IMS addresses the

challenges of service interdependence, where simple service building blocks are combined to deliver more sophisticated end-user services that integrate multimedia, data and voice within a single user session. The openness and distributed nature of the marketplace will generate a new wave of innovative services by enabling the technology and business capabilities of service providers, still providing benefits from any actor in the value network, including operators, infrastructure vendors, content providers and others. Presence and location-based services are just a few of the exciting and potentially lucrative service enablers that are delivered with IMS. These service enablers are then combined with voice, value-added text and multimedia within a single user session to deliver simple, engaging and easy-to-use IMS end-user services.

Several researchers believe that IMS standards will stabilize in 2010 (Research, 2008), as by then it is anticipated that standards bodies, platform vendors and service providers will reach a critical-mass stage of consensus around architectures, security and broad service categories. By 2010, market growth will see an inflection point and the contribution of IMS gear revenues to the overall equipment market will ramp up as barriers to adoption fall. Therefore, most IMS services will be successfully integrated in the digital marketplace.

6.2 Developers and Service Creation

The convergence between the telecom world and the Internet is one of the engines of innovation nowadays. As discussed previously, many operators are launching initiatives to open their capabilities towards the web world, and there are an increasing number of developers expecting interesting features to build new services with features of both worlds. At this point, technology is crucial. Operators should offer their functionalities in a friendly way to developers and the latter should use the appropriate programming languages and frameworks to create applications that are mostly coming from the Internet web world. This section picks some of the key representative technologies of this new wave for developers and service creation. Examples on these are, for instance, REST as the paradigm for opening a service with distributed APIs that is reachable from any point on the Internet; Ruby on Rails as a representative framework for accelerating fast development of applications with minimum effort and maximum reusability of code; rich Internet application techniques and flavours as boosters of the creation of appealing applications with easy-to-use human interfaces; and AJAX as the interim step towards the power of RIAs, providing a set of combined mature technologies that together have consolidated the web as an online user interface with unprecedented interactivity and dynamicity.

All these technologies that are coming from the new web world are expected to be applied to telecom–web converged services and to the 'telecom-only' applications as well, as they have proven their convenience in easing the developers' task and for fostering creativity of service designers. The true multimedia experience that IMS services render to users is a perfect companion of these interactive-rich multimedia applications that can be built under the combination of RIA, AJAX and REST techniques. In fact, most of the technologies that have been presented in this section are perfect candidates for the realization of most of the service strategies pursued by WIMS 2.0 and presented in Chapter 5. For instance, RIAs and AJAX enable the technical realization of the IMS virtual client in the desktop, web and mobile environments and hence seamlessly extend the user's perception

of the telecom services from the native client residing in telecom-specific devices, like mobile phones and IPTV set-top boxes. REST, as a key paradigm for a new-generation exposure layer of telecom services and capabilities has already been discussed in Chapter 5.

6.2.1 Opening Capabilities: REST

Representational state transfer (REST) is a term coined by Roy Fielding in his PhD dissertation (Fielding, 2000) to describe an architecture style of networked systems. The World Wide Web is the key example of RESTful design. REST is intended to evoke an image of how a well-designed web application behaves: a network of web pages (a virtual state machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use. An important concept in REST is the existence of resources, each of which is referenced with a global identifier (e.g. a URI). In order to manipulate these resources, components of the network (clients and servers) communicate via a standardized interface, e.g. HTTP, and exchange representations of these resources (the actual documents conveying the information). An application can interact with a resource by knowing two things: the identifier of the resource and the action required. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML, XML or JSON document of some kind, although it may be an image, plain text or any other content.

6.2.2 Application Framework: Ruby on Rails

Ruby on Rails is an open source web application framework for the Ruby programming language that uses the model view controller (MVC) architecture pattern to organize application programming. It combines simplicity with the possibility of developing real-world applications writing less code than other programming languages. It is intended to be used with the agile development methodology, which is often utilized by web developers for its suitability for short, client-driven projects. Rails initially utilized lightweight SOAP for web services, which was later replaced by RESTful web services. Since version 2.0, Ruby on Rails by default offers both HTML and XML as output formats. The main principles of Ruby on Rails include 'Convention over Configuration' (CoC) and 'Don't Repeat Yourself' (DRY). 'Convention over Configuration' means a developer only needs to specify unconventional aspects of the application. Generally, this leads to less code and less repetition. 'Don't Repeat Yourself' means that information is located in a single, unambiguous place. For example, using the 'ActiveRecord' module of Rails, the developer does not need to specify database column names in class definitions. Instead, Ruby on Rails can retrieve this information from the database.

6.2.3 Development Techniques: AJAX

One of the main technologies in Web 2.0 is AJAX (asynchronous JavaScript and XML) (Gittleman, 2006). AJAX uses a last generation browser and allows more interaction in applications development with the advantage of not being needed to install anything in

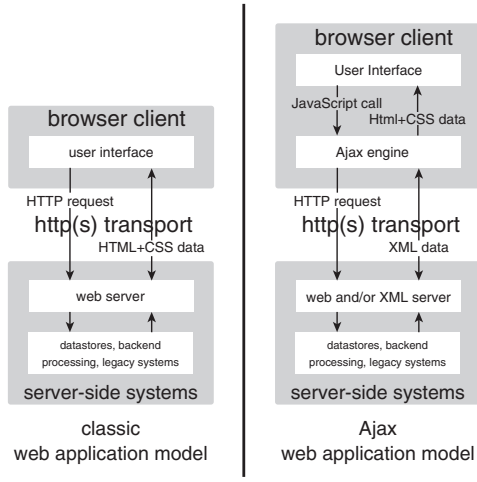

Figure 6.5 The AJAX model.

the client (see Figure 6.5). AJAX breaks the bad interactions from the request–response HTML forms, introducing an asynchronous request to the server with responses with partial reloads of the same web page, seeming a high interactive application and enhancing the user experience. AJAX has existed since the end of 1990s, but it is only now, with new browsers and more bandwidth, that it has been spread. Unlike other client technologies such as Java or Flash, the server keeps the application and interface status, not requiring an initial phase of application loading. It is based on standard web technologies (JavaScript, XML y HTTP), although not all browsers have the adequate support.

The main features are:

- AJAX allows building applications, rather than websites, enabling features to be developed that previously were out of the scope of the applications.
- Continuous interaction, similar to what the user is used to in common applications.
- Visual and interaction richness. Despite the name, the use of JavaScript and XML are not actually required and nor do the requests need to be asynchronous. AJAX is a valid technique used in multiple platforms and operative systems because of being based on open standards, namely:
- HTML and cascading style sheets (CSS) for presentation.
- Document object model (DOM) accessed with a scripting language by the user, specially ECMAScript implementations like JavaScript and JScript, for dynamic display of and interaction with data.
- XMLHttpRequest object for asynchronous communication.
- XML is the format used for the data transfer requested to the server, although any format can work.

With this technology, problems such as application distribution or the distribution of new versions or patches is solved, thanks to the use of a light client: the browser, which becomes a client-side platform container of service logic. The impact of these applications

has faded the distinction between web and desktop applications, being a very interesting alternative for developers.

6.2.4 Rich Internet Applications

A rich Internet application (RIA) (Wikipedia, 2009) is a web application designed to deliver the same features and functions normally associated with desktop applications. Applications are rich or enriched by their ability to integrate multimedia capabilities such as sound, animation and video in addition to a greater degree of interactivity between the application and the client's machine. The user can thus instantaneously interact with the application without having to wait for roundtrips to and from the server because the entire application is loaded from the start. RIAs provide true two-way communication between clients and applications, moving away from the traditional page-to-page web navigation model. An RIA normally runs inside a web browser and usually does not require software installation on the client side to work. However, some RIAs may only work properly with one or more specific browsers. For security purposes, most RIAs run their client portions within a special isolated area of the client desktop called a sandbox. The sandbox limits visibility and access to the files and operating system on the client to the application server on the other side of the connection.

The development of enriched applications for the Internet responds to a necessity of elevating presentation levels (front end) so that they have the ease and richness that users expect. RIAs focus on increasing the optimization of the final user's experience and the reduction of the bandwidth and the server's load. There are some examples of RIA frameworks and technologies from different creators. In the remaining part of the section the Adobe, Microsoft and Google approaches on RIA are discussed.

6.2.4.1 Adobe AIR

Launched in early 2008 as the 1.0 version, the Adobe integrated run-time (AIR) is an interesting development framework that enables web designers and developers to use the technologies used to build websites to build applications that are used on the desktop. This implies technologies like Adobe Flex, Adobe Flash, HTML, JavaScript or AJAX. Adobe AIR runs on Windows, Mac OS X and recently on Linux. A version for mobile devices is under development.

A web browser enables a user to interact with the content and applications typically located on a website on a server. Adobe AIR builds upon capabilities and technologies used in the browser to enable deployment of applications on the desktop. Adobe AIR complements the browser by providing users and developers with a choice about how to deliver and use applications built with web technologies. Because they run on the desktop and not in a web browser they can offer more features and access and save files on the local hard drive or network. It can interact with online resources if connected or use local storage (via the included SQLite database) and synchronize data with an online source when next connected. Adobe AIR is also smart enough to know when it is and is not connected so you can create programmes that work with online data and/or use the local database as a fallback if the connection to the Internet is dropped. When compared to traditional desktop applications, Adobe AIR applications are simple to deploy, easy and cost-effective to build, have better web integration and will run on all three of the major operating systems.

6.2.4.2 Adobe Flash

This is a multimedia authoring and playback system from Adobe used to create RIAs. Adobe Flash in commonly used to create content such as web applications, animations, advertisements, games, movies and applications that accepts input from the user. Flash contents are also available for mobile phones and other embedded devices. ActionScript is the language used to write Flash programs. Several software products, systems and devices are able to create or display Flash contents, including Adobe Flash Player, a free client application which is available for most common web browsers, some mobile phones and other electronic devices (using Flash Lite). The files created using Flash are Shockwave Flash (SWF), played in Flash Player, and Flash Video (FLV), played in VLC, QuickTime and Windows Media Player with external codecs added. Flash content reaches over 98% of Internet viewers with over 30M SWF files online today. Moreover, around 70% of web video is in Flash.

On May 2008, Adobe announced the Open Screen Project, which aims to drive rich Internet experiences and create a consistent application interface across all devices like personal computers, mobile devices or consumer electronics. It was perceived that the main goal of the project is to bring Adobe Flash to mobile phones. A project advantage is that it abolishes licensing fees for Adobe Flash Player and Adobe AIR. It also removes restrictions on the use of the Shockwave Flash (SWF) and Flash Video (FLV) file formats and publishes application programming interfaces for porting Flash to new devices. Additionally, it publishes Flash Cast protocol and Action Message Format (AMF), which allows Flash applications to receive information from remote databases.

6.2.4.3 Adobe Flex

At the present time Adobe Flex version 3 is a collection of technologies released by Adobe for the development and deployment of cross-platform RIAs based on the proprietary Adobe Flash platform. Adobe Flex is a free, open-source framework for building highly interactive, expressive web applications that deploy consistently on all major browsers, desktops and operating systems. MXML, a declarative XML-based language, is used to describe graphic user interface layouts and behaviours, and ActionScript 3, a powerful object-oriented programming language, is used to create client logic. RIAs created with Flex can run in the browser using Adobe Flash Player software or on the desktop on Adobe AIR, the cross-operating system run-time. This enables Flex applications to run consistently across all major browsers and on the desktop applications, reaching more potential users. By using AIR, Flex applications can now access local data and system resources on the desktop. Since February 2008, Flex 3 is available as an open-source software through the Open-Source Flex SDK Project. Flex provides a modern, standards-based language and programming model supporting common design patterns.

6.2.4.4 Google Gears (Google, 2009b)

Created in 2007, Google Gears is free and open-source software offered by Google. Gears is a plug-in that extends the browser to create a richer platform for web applications. The broader goal is to close the gap between web applications and native applications by giving the browser new capabilities. For example, webmasters can use Gears on their websites

to let users access information offline or provide contents based on customer geographical location.

Gears installs a database module, powered by SQLite, in the system client that stores data locally. Therefore a web application can synchronize the local data with the online service. Stored local information is mainly used instead of online information. Synchronization can wait until a connection is available or until the user wants to do it. Gears also has a desktop module that allows web applications to interact more naturally with the desktop and a geolocation module that lets web applications detect the geographical location of their users. As for now, Gears is compatible with Internet Explorer, Mozilla Firefox and Safari browsers. Gears was designed to be used on both Google and non-Google sites. Examples of web applications that currently make use of Gears are Google Reader and Google Docs.

6.2.4.5 Microsoft SilverLight (Microsoft, 2009b)

This is a cross-browser, cross-platform implementation of the .NET framework for building media experiences and rich interactive applications for the web. Version 2.0 brings additional interactivity features and support for .NET languages and development tools. It is compatible with multiple web browser products used on Microsoft Windows and Mac OS X operating systems. Mobile devices, starting with Windows Mobile 6 and Symbian (Series 60) phones, will also be supported. A third party free software implementation named Moonlight is under development to bring compatible functionality to GNU/Linux. SilverLight supports the display of high definition video files so that Microsoft will do the heavy lifting of sending them over the net. This will enable smaller developers to deliver large and high definition files quickly and reliably without paying for content distribution network fees. Also, SilverLight applications are delivered to a browser in a text-based markup language called XAML. That is not difficult for web users once they land on a site. However, search engines, like Google, can scan XAML but they cannot dive into compiled Flash applications.

6.3 Users and Service Creation

One of the promises of the future Internet of services is to bring a huge amount of functionalities that can be combined easily to develop new applications or services that are more customized to the user premises. With an approximate rate of one new open API per day, the number combinations of those APIs to create new mash-ups is going up dramatically. Nowadays, it is easy to create mash-ups, but a knowledge of programming languages is still required to compose the new application, which limits the access of any user. Nevertheless, developers act in two main lines: creating applications based on common interests that can satisfy a group of potential users under demand or creating applications to satisfy some of the developer's necessities. Going to a more and more customized world, it still seems difficult to see a developer creating one application for only one user, targeted on the user's necessities or interests. However, what could be quite probable is to see one user developing applications to satisfy his or her own necessities if there are some tools to make this situation possible and easy. In fact, user-generated services (UGS) deal with this strategy. The goal is to create some tools that allow easy creation of a new service by only being familiar with the environment and the APIs available in it. Most of those tools

are using click-and-drag functionalities for achieving such simplicity of use. Tools to create UGS are appearing on the market from different providers: Microsoft, Yahoo, Google, etc. In this section, Internet-oriented UGS tools like those by Yahoo and Microsoft are prenetted together with a telco-oriented solution coming from the European research space: OPUCE. Google also has a platform called Google Mashups Editor, but it still remains in close beta version limited and hence to a small number of developers.

6.3.1 UGS Based on Data Mash-ups: Yahoo Pipes

Yahoo Pipes (Yahoo, 2009) is the current UGS solution from Yahoo that provides a graphical user interface for building applications that aggregate web feeds, web pages and other services, creating web-based applications from various sources and publishing them (see Figure 6.6). The site works by letting users 'pipe' information from different sources and then set up rules that establish how the content is modified or retained (e.g. filtering). A typical example is a WIMS 2.0 case based on Flickr – a pipe that takes any RSS feed and adds a photo from Flickr based on the keywords of each item of the RSS stream. In this case, Yahoo Pipes redirects flows of data, a chained flow of actions, where the result of one serves as input for the next.

Pipes provides huge possibilities to handle and manipulate the content on feeds, translate them and filter for keywords, creating personalized filters. The pipes editor is a JavaScript

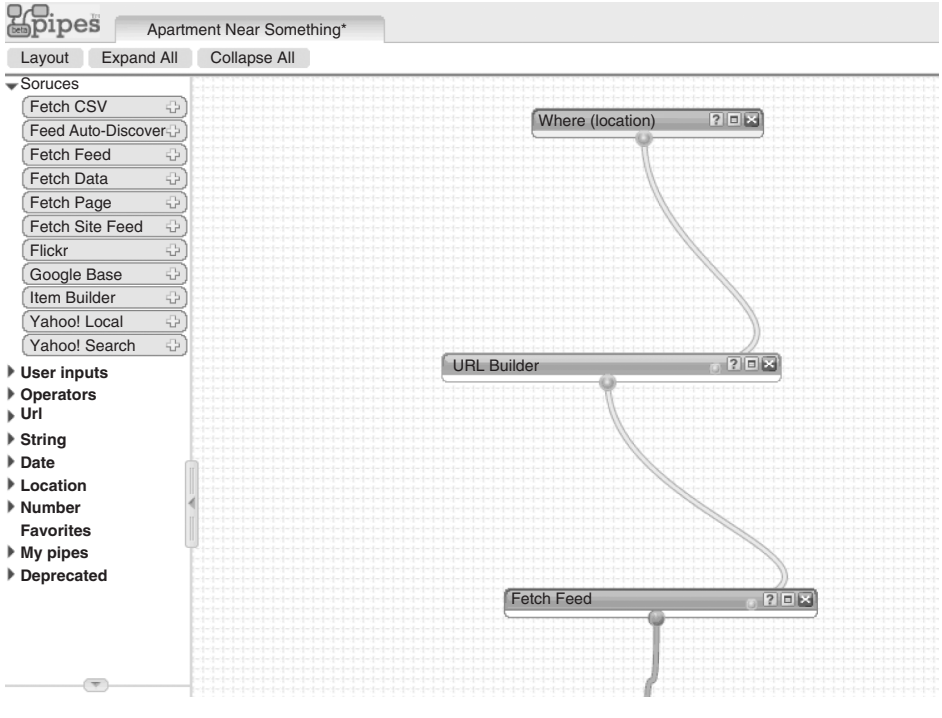

Figure 6.6 Yahoo Pipes.

tool that lets users create and edit pipes in an intuitive visual interface. The editor consists of the following pieces:

- Library, which lists available modules and saved pipes;
- Canvas, which is the main work area for assembling pipes;
- Debugger, which lets a user inspect pipe output at various stages.

A user can build and edit pipes by moving modules on to the Canvas from the Library and wiring them together.

6.3.1.1 Library

Library shows a list of all the available modules, which are grouped by functionality:

- sources: data sources that return an RSS feed;
- user inputs: input fields that a user of the pipe can fill in at run-time;
- operators: basic features like foreach, sort, count and filter;
- URL: modules for building and manipulating URLs;
- string: modules for handling strings;
- date: modules for manipulating dates.

The Library model also holds a list of user's favourite pipes belonging to other users and user saved pipes. Users can build new pipes by using those created previously by the same or other users, constituting these subpipes as new building blocks. This enables users to build useful complex components that can be reused across multiple projects. For the sake of simplicity, which is guaranteed in all aspects of the pipes–user interface, drag-and-drop functionality is enabled in order to add a module or subpipe to a new pipe and the user only has to drag it over to the canvas.

6.3.1.2 Canvas

The Canvas is the main work area for assembling and testing new pipes. The user can drag modules around and arrange them based on user decisions or using predefined layouts in the tool. The user has to wire modules together by clicking the output terminal of any module and then clicking on the input terminal of the module to be fed with data from the generator module. The editor will check the compatibility of the terminals (what kind of data that a terminal expects to emit or receive), showing them in orange to the user. Many modules have configurable parameters and input fields that can be filled by the user or supplied with information from another module.

6.3.1.3 Debugger

This shows the contents of the pipe immediately downstream from the currently selected module, allowing the inspection of each segment of the pipe to make sure it is behaving as expected. Once the pipe has finished, it can be published, making it available to the wide Internet. This case will allow people to clone a copy for their own use.

6.3.2 UGS Based on RIA: Microsoft Popfly

Microsoft Popfly (Microsoft, 2009a) is a tool that allows users to create web pages, program snippets and mash-ups using the Microsoft Silverlight (Microsoft, 2009b) RIA (rich Internet application) run-time platform and the set of online tools provided with it. It requires users to log on with their Windows Live ID that is the Microsoft commitment to provide a unique identifier. Popfly provides four tools based on SilverLight technology:

- Game Creator. A tool that allows users to create their own game or extend a game already built. It can be exported to the social network Facebook or be used as a Windows Live Gadget.
- Mash-up Creator. A tool that lets users fit together pre-built blocks in order to mash together different web services and visualization tools.

For example, a user could snap together a photo building block and a map block in order to get a geo-tagged map of pictures on a topic of their choice. Also Popfly presents an advanced view for blocks, which allows users to modify the code of the block in JavaScript, as well as providing users with flexibility in designing the mash-ups. Additional HTML code can also be added to the mash-ups. This tool also provides a function for auto-completion of HTML code and a preview function, with a live preview in the background as users link blocks.

6.3.2.1 Web Creator

This is a tool for creating web pages. Web pages are created without HTML coding and can be customized by choosing predefined themes, styles and colour schemes. Users can embed their shared mash-ups in the web page. Completed web pages will also be saved in each user's Popfly Space. The Popfly Space provides a quota of 100 MB per user where completed mash-ups and web pages are stored. Users also receive a customizable profile page and other social networking features. Public projects can be shared, rated or 'ripped' by other users. Popfly allows users to download mash-ups as gadgets for Windows Sidebar or embed them into Windows Live Spaces. Another feature of Popfly Space is the Popfly Explorer plug-in for Visual Studio Express to download the mash-ups and modify the coding, as well as perform actions such as uploading, sharing, ripping and rating the mash-ups. Popfly is more powerful than Yahoo Pipes, which is limited to RSS channels, as previously mentioned, enabling users to build mash-ups in minutes without any knowledge of either programming or writing a line of code. These mash-ups can be downloaded as gadget, iframe or added to the Popfly Space or some social network sites such as Facebook. One of the existing limitations, however, is in the source blocks, where a user finds Live resources and some third parties resources (Yahoo, Twitter, etc.). It is difficult to see some interesting resources from Google. Popfly allows users to create their own blocks, but it is not possible to choose sources not considered by Microsoft. Popfly interface is usable and powerful and it is focused on amateur developers and enthusiastic people creating new applications.

6.3.3 UGS Based on Telecom Convergence: OPUCE

OPUCE (open platform for user-centric service creation and execution) (OPUCE, 2009) aims to foster the evolution from the absolutely successful 'user-generated content' digital society

to a 'user-generated services' one. OPUCE brings Web 2.0 benefits to telecommunications by providing tools that allow nonskilled users to build, share and control their own personalized convergent services, combining both web and communication capabilities. In brief, OPUCE leverages on WIMS 2.0 strategies for service convergence by adding UGS facilities and mechanisms to the WIMS 2.0 ecosystem. Some of the most remarkable advantages of OPUCE are:

- New convergent services. This covers web applications and telecommunication services (e.g. voice calls, SMS, MMS, etc.) interaction. Users will gain a real global experience, increasing satisfaction.
- End-users and customers involvement. End-users can participate fully in the creation and development process. This allows fast and early access to new services, generating a thriving space for service creation, adjusted to real market and user demands, and reducing services-associated costs.

Nowadays, there are several web mash-up solutions available, as mentioned earlier (Pipes or Popfly), but they have some strong drawbacks, such as the fact that they are only focused on combining Internet services and their fields of application are limited or unsuitable for professional usage. OPUCE is designed for users without programming skills, enabling them to mix Internet and telco capabilities in order to compose services to leverage traditional communication system constraints. Moreover, it offers smart solutions with a drag-and-drop graphical user interface, which allows a more engaging interaction for service creation and consumption, increasing user satisfaction. Figure 6.7 shows a comparison of the existing UGS platforms, with a detailed comparative description on fine grained features.

OPUCE considers three roles for interacting with the platform:

- Service Creator is the user who logs on the platform to create a new service combining base services, known as atomic enablers.
- Service Subscriber is the one who, after browsing a service catalogue or being notified when a new service of his interest has been created, can subscribe to it and can start using the service.
- Service Administrator has privileges to manage the OPUCE platform, being able to alter the lifecycle of a service or removing services that are no longer needed.

One of the distinctive features of OPUCE is the support of ubiquitous access to different applications from different end-user devices (PCs, classic telephones, mobile phones, PDA, etc.) via different wired and wireless networking technologies, at each stage of the service lifecycle including execution and creation. Figure 6.8 shows that OPUCE is positioned as a platform with two main features: simplicity and remixability between the telco world and the web.

OPUCE definitely enhances typical mash-up tools with telco-specific services support, including telco resources such as provisioning, delivery and management mechanisms. At the same time, it intends to be a pluggable solution that can be seamlessly integrated on top of existing telco service delivery infrastructures and open interfaces, by providing additional tools that greatly simplify the management and use of such environments, enhancing user satisfaction and quality of experience. In that sense, OPUCE complements the WIMS 2.0 so that the latter provides the foundation, i.e. the open service architecture and exposure layer of telecom service capabilities, to be integrated into the OPUCE editor.

OPUCE features		Web 2.0 Mashup tools						Telecom providers solutions					others	
	OPUCE	Popfly	Yahoo! Pipes	Google mashup editor	Microsoft Connected Services SANDBOX	Developer Portal	BROADSOFT XTENDED	openkapow	Web2Click	betavine	Bubbletop	Dummy	ActiveVos	EzWeb
Telco services	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓
IT services	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Messaging	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗
Presence	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗
Voice calls	✓	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗
Audio/video conferences	✓	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗
Authentication	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓
SMS	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗	✗
Call control	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗
Easy services composition	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓
Web Editor	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓
Mobile Editor	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
content creation	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓
Events oriented	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
clean service specification	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
service logic ("IF..." blocks)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
User privileges / Rights management	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓
Suscription to notifications	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
New services notifications	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	✓	✗	✗
context adaptation	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Full Service Lifecycle Mgmt.	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Open to 3rd parties	✓	✓	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✗	✓
User's community	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Runtime configuration	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Personalization based on user profile (Sme)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Tools for "building blocks" (Base services) creation	✓	✗	✗	✗	✗	✓	✓	✗	✗	✓	✗	✓	✗	✓
Mobile Portal	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗
Gad gets widgets	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
View of the code generated	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗
Suggest to components to connect to	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
View the results of the execution in the creation tool	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Direct execution on the browser (no backend)	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Designing tools	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗

Figure 6.7 A comparison of the most relevant UGS platforms.

The OPUCE platform can be described in terms of the functionalities it provides to users, or the elements that conform to the platform and support those functionalities. Firstly, the OPUCE functionalities are described below:

- Service creation. An advanced service creation environment allows end-users to create easily and intuitively new services combining building blocks to control the workflow among them. This environment is available both for web and mobile users and enables them to cross-edit services (see Figure 6.9).
- Service description. Services are described in a multifaceted extensible approach where each facet describes a specific concern of the service: functional features, nonfunctional features and management features.
- Service lifecycle management. A feature to control and manage the whole service lifecycle from creation to withdrawal of the services through a set of components such as service deployment, service repository, service provisioning and service monitoring.
- Service execution. A functionality to provide a highly scalable and fault-tolerant hosting environment for running services.

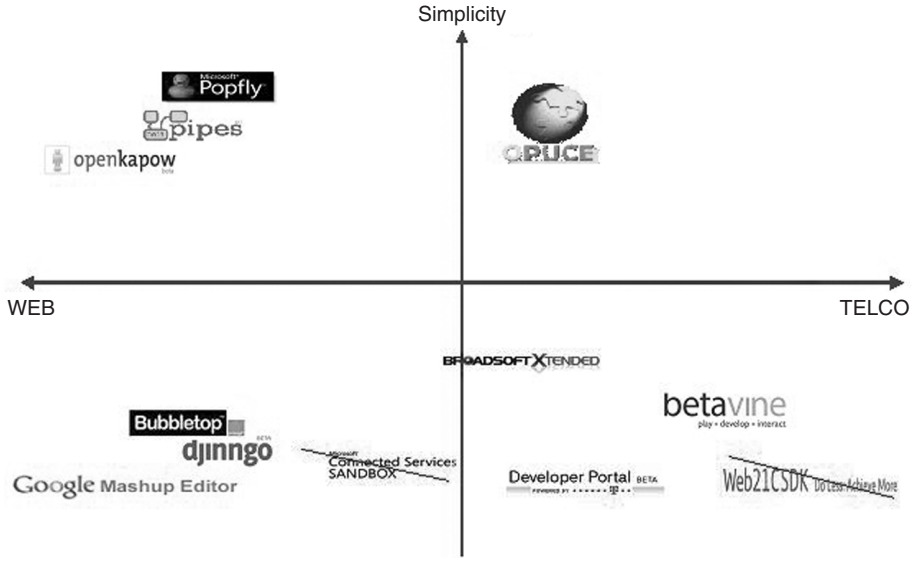

Figure 6.8 Positioning of the main UGS platforms.

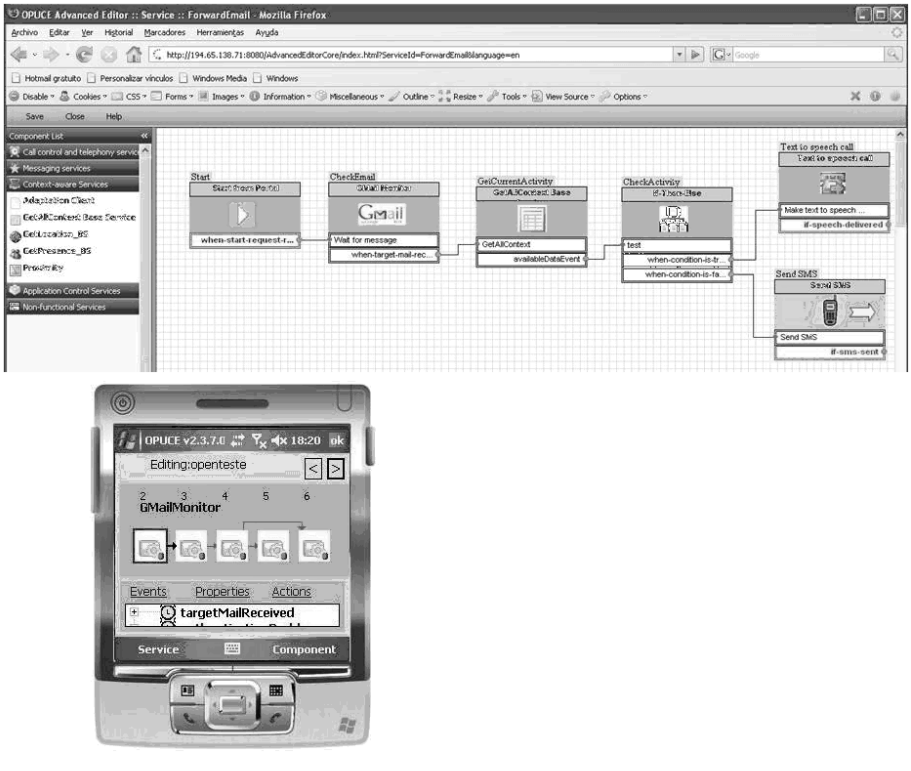

Figure 6.9 OPUCE web and mobile editor.

- Service advertising. An advanced service advertising system with learning engine and service recommendation will be built to facilitate the sharing of services between various levels of system participants – from operators to end-users.
- Service adaptability and personalization. The platform must be aware of user's profile parameter changes and must be capable of adapting services accordingly with those changes. The OPUCE platform is constituted of the following elements.
- Portal. This is used to allow service creators to compose new services from base services available and to contribute to the creation of service building blocks to be shared in a common repository. Through the portal, users can be managed by service administrators and service subscriptions directly by the users themselves.
- Service lifecycle manager. This provides interfaces to deploy and provision services, allowing them to be available automatically to service subscribers depending on its ambience, for instance. At run-time, the service lifecycle management is used to monitor execution characteristics of running services and tune their run-time parameters and deployment configuration.
- Service execution unit. This provides a distributed, highly scalable, fault-tolerant environment for running services.
- User information manager. This stores in a secured way critical and sensitive information about users, such as identity, profile, preferences and context.
- Service advertising unit. This allows users to share services and be notified about services using multiples channels. It also features an intelligent learning engine to match user characteristics and preferences to specific service features.
- Context awareness unit. This is used to personalize and adapt service executions based on certain user conditions (device features, media capabilities, ambience and user interface capabilities).

6.4 Devices and Service Creation

Applications development in mobile devices has suffered, from the beginning, from a huge fragmentation and an added complexity. While at PC platforms, implementation details were hidden by operating systems while in the mobile world the situation was not the same. Moreover, most of the initiatives were born with a not-so-open model as everyone was trying to maximize the return on investment by capturing a high market share. Innovation and development of applications for mobile devices has evolved slower than in the fixed world due to the necessity to modify each application for the different mobile device models and the limitations of those devices. There is no unified mobile device platform for easing development of applications. Instead, different aspects like screen resolution, APIs availability and configuration details have to be modified from one device to another. This causes an application developed for some of these mobile platforms not to work properly in the rest, hence increasing the price and the complexity of the application and building a barrier for developers. Likewise, any modification of the application requires the same arduous way, which results in disillusioning developers to evolve and enhance applications. Furthermore, given the targets for market penetration, an operator does not risk launching a new service unless it is supported by all relevant parts of the terminal park under its control. As network capabilities evolve, the intelligence required at the terminal side is usually higher for most innovative applications. This leads to an increasing need to deploy application logic at the

terminal side in order to get a service in place. For instance, most of the innovation that evolved in telecom networks, such as those foreseen by an IMS-enabled telecom network, is based on the fact that new applications can be developed by distributing service logic at the network side and the terminal. An example is an IMS-enabled rich communicator with enhanced user interfaces or IMS-based games. This indeed requires developing client applications for mobile device platforms.

The current functionalities of mobile phones (computing speed, screen resolution, colour width in displays or Internet access speed) are similar to the functionalities of PCs ten years ago. The abstractions done by handsets manufacturers over the existing hardware are minimal in order to get the highest speed and the best real functionalities of each device. On the other hand, real efforts from manufacturers to merge with a common platform are short. The increase in smart phone devices allows similar development environments to be accessed to the PC/UNIX, but still there is great fragmentation and many limitations such as the high degree of specialization required to develop applications or the use of costly and proprietary development environments.

Before explaining future trends, it is necessary to mention the Java (J2ME) solution. A promising solution of the slogan 'develop one – run many' was hype. The reality is that to develop an application requires adaptation and tests for each of the handsets models where the application will run, not to solve existing problems.

6.4.1 Mobile Browser: Web-Based Applications

When examining terminal platform possibilities for creation and development of services overcoming multidevice challenges, the web browser appears to be a good candidate for developing cross-terminal applications by utilizing the web environment in the mobile device: the web browser. The Web 2.0 revolution has promoted web development techniques specifically targeted at adding functionality at the client side, like AJAX, described in the previous section.

The term AJAX has come to represent a broad group of web technologies that can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. With the advantage of no requirement to install anything in the client, it uses the last generation browser. With this technology, problems such as application distribution or new versions or patches distribution are solved, thanks to the use of a light client: the browser. Mobile devices include browsing applications with varying features and complexity. A homogenization in this respect is a trend that has been happening since 2007, where expectations are that most of the 80% of feature phones in the market in 2010 will support AJAX and other Web 2.0 techniques.

6.4.2 Mobile Widgets/Gadgets

One of the tendencies inside the current web is to build applications that are developed and run in an application container. The use of preconfigured application frameworks and a remix of information allow this kind of application to be developed in a short time. These frameworks render the information and service presentation aspects at the mobile platform side, but the logic of the application runs in a remote container. The complexity lies in

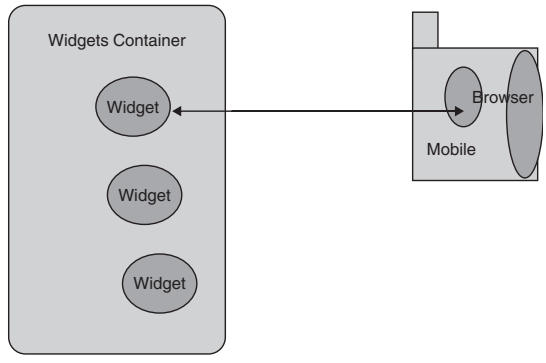

Figure 6.10 Widgets container.

choosing the right technology. Each of the manufacturers and ISPs has a version, which are seemingly the same but are usually incompatible. This is also the case with the so-called ODP (on-device portal), which is offered by most mobile handset manufacturers to the operators, where a small Java application containing the different widgets is installed in the mobile device (Figure 6.10). When the user runs this application a connection to a server is established in order to run the widget's functions. The problem here is the fragmentation due to the lack of interoperability between manufacturers. In order to offer common solution thinking to the user, different working groups in the World Wide Web Consortium (W3C) are working to develop a technology and a set of standards that allows interoperability between platforms.

If these applications are embedded in any web page and also run, without changes, in the mobile device, the problem of a unique user experience with applications and services designed easily at one time and run in many different environments would be solved. In parallel, one of the goals for these applications is to run 'offline' for those moments when it is not possible to have an adequate mobile coverage level, ensuring at least a minimum performance.

6.4.3 Mobile RIA Applications

Flash and Adobe technologies have evolved towards a simple solution, although proprietary, to develop interactive applications over a browser. The Adobe's challenge is to start a version of this software on ARM processors that would constitute a very adequate alternative to develop mobile applications due to the huge number of developers able to use this technology. Many current browsers have Flashlite, a reduced version of the Flash environment with fewer capabilities, but it still is not clear in which devices these will run. Finally, there is a way to run these Flash/FLEX applications as another application in the operating system, using AIR, although this platform is only available for PCs. Adobe's major competitor for mobile RIA platforms is Microsoft, with SilverLight, a more recent framework that will probably be catering for niches such as mobility. Microsoft plans for SilverLight to support mobile devices, starting with Windows Mobile and Symbian OS.

6.4.4 Android Environment

The Android environment proposed by Google (2008c) has the objective to increase advertisement and user profiling business by increasing the number of devices connected to the Internet, with the inclusion of mobile devices in the mesh. Google proposes to offer the same user experience already offered in PC environments. That is the 'look-and-feel' combined with the set of services and integration of all the pieces that Google provides. The main problem in mobile devices is the high fragmentation in the existing operating system offering that requires adaptation of applications on a case-by-case basis, resulting in different user experiences for a common application. To avoid this situation, Google has created Android: a flexible, solid and free execution environment clearly focused on the user. The main goal is to have the same user experience for the applications independent of the access. Android works in two lines:

- To offer the best user experience, unified and attractive, independent of the devices.
- To reduce the production cost of a device with android up to 10%. In parallel with the development of Android, Google has built an ecosystem via the Open Handset Alliance (OHA:www.openhandsetalliance.com), which is based, at least initially, on creating win–win relationships between the members, each one pursuing a specific interest:
  - Device manufacturers: reducing costs in licenses and TTM due to the fact that Android is easily integrated into manufacturer's hardware.
  - Developers and content providers: making a reality from the paradigm 'write once, run many'.
  - Operators: an open environment to personalize and integrate services quickly and easily.

In order to ensure the success of Android, there is a need to have a critical mass of handsets selling to feed the ecosystem. It seems that Nokia, focused on the S40 and S60 series, or Sony-Ericsson and RIM, with proprietary platforms, are not viable candidates to play in this 'game'. HTC implemented the first Android mobile phone, the G1, launched commercially by T-Mobile in 2008 (Google, 2008a). The rest of manufacturers, such as Samsung, Motorola or LG, could be candidates to implement Android inside their devices. The high level architecture of the Android platform is represented in Figure 6.11.

Based on the target requirements, Android presents the following features:

- Execution environment completely developed from J2ME.
- License model: kernel under GPLv2, user-space under Apache.
- 'Dalvik' virtual machine: executes code optimized for mobile devices (.dex) and manages memory more efficiently.
- HW requirements: processor ARM9 – 200 MHz, 128 MB RAM, mini/micro SD, screen QVGA TFT 16 bits. Optionally required is Qwerty keyboard, WiFi or GPS.
- SW requirements: Linux 2.6.
- Browser: based on KHTML, WebKit. Full-navigation, CSS, javascript, DOM, AJAX.
- Graphics: SGL for 2D and Open GL for 3D.
- Media framework: based on open code from PacketVideo.

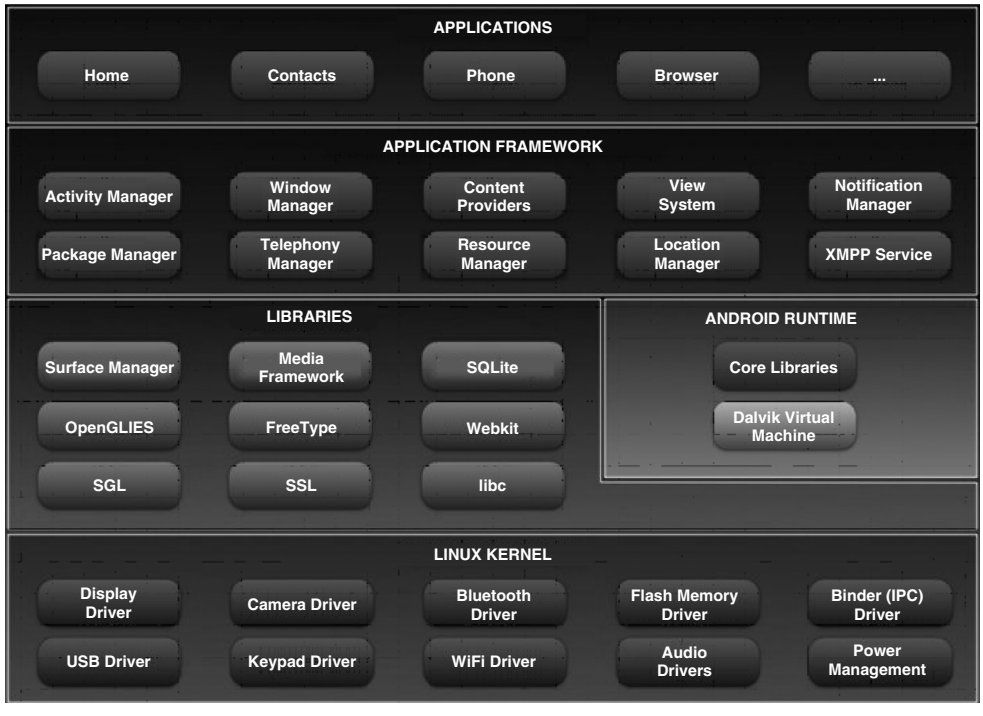

Figure 6.11 The Android architecture.

- Security model: applications have to declare the access to sensitive resources in order for the end-user to authorize the access in execution time.
- Specific APIs for Google services: GMaps, Media and XMPP.

The Android SDK allows the development of applications in Java and is perfectly integrated with Eclipse, although it has other additional tools. Android redefines the lifecycle of an application, delegating its own management process to Linux. Each application runs in a process, thus ensuring the system robustness. The design of the interfaces is based on XML with the objective to ensure compatibility between devices with different graphics capabilities. The use of XML enables the enhancement of the performance obtained using AJAX (Galindo, 2007).

In summary, Android provides a modular, robust and, what is more important, open framework based on Linux, which allows the interest of a much larger community to be captured. Android provides a tool for an environment with high fragmentation and difficult access for application developers. Android is open, offering opportunities for environment personalization never seen before in the mobile world. Nevertheless, there are open questions to answer in the future, such as the paradigm to develop an application that works in many heterogeneous handsets or to have the 'Google experience' homogeneously in any device, fixed or mobile. Considering the business aspects, Google is not only sure of its objective but also, for this adventure, knows that it is needed to create a right ecosystem.

6.4.5 The iPhone Toolkit

Apple designed the iPhone applications strategy based on AJAX applications, but instead of installing any software in the mobile device and after evaluating the low impact on the market, Apple decided to open it using a similar toolkit to the OSX.

The SDK is based on Objective C. The graphical and widgets toolkits are based on Cocoa and Carbon, allowing developers the quick creation of applications. The development platform that comes with many facilities already available to easy developments consists of four modules:

- Core OS is based on the OS X kernel and includes functions for power management, library system, security systems, sockets and network protocols implementation.
- Media for multimedia and graphics management. To highlight the support of 3D graphics using OpenGL ES and 3D positional audio, with open AL and Core Animation for animation creation.
- Core services provide access to files, threads management, network services management, preferences, address book, URLs, process and Core Location and geolocation management.
- Cocoa Touch, which is the framework to create tactile interfaces, enables a set of classes to handle events and controls (Multi-Touch, accelerometer, view hierarchy, localization, alerts, web view, people picker, image picker, camera).

Developers can have access to the same tools used internally by Apple for the software development for the iPhone. For instance, XCode is the integrated development environment with debugging support, which is the same as that used in Mac. Interface Builder is used to design application interfaces and to join the interface elements with the code based on a drag-and-drop tool. The iPhone simulator allows the running of the applications developed at Mac OS X without an iPhone handset. This tool also enables a debugger function of the applications running in the iPhone connected to the Mac. Instruments is a set of tools to check the application performance while they are running in the iPhone. For those developers programming in Mac, it is easy to create an application for the iPhone.

The installation model is based on iTunes with centralized and verified download during the connection. This model, says Apple, avoids malicious applications installation, but also is an income for Apple if the developer is planning to sell the application.

One important limitation, however, is that not all Apple applications are running in the background, which means that there is a need to save the status of each application and no asynchronicity is naturally possible. The iPhone core is the same as that of Mac. The iPhone is not a mobile phone with multimedia facilities, but a multimedia device that includes a telephone facility.

6.4.6 Maemo

Although it is not focused on mobile phones, but rather in tablets PC with WiFi connectivity, it is quite probable that Nokia is considering introducing this software in its top range of mobile phones, so-called ‘smart phones’. It is based on a special distribution of Linux, called Maemo (Maemo, 2008), which will enable opening mobile devices to all existing Linux

applications. It is a version of SuSe Linux (SuSe, 2008) adapted to the ARM processor, which is currently included in smart phones.

It is a very promising line, but it has to be developed for a set of mobile handsets that still neither exits nor have been announced. Maemo enables a quick development in this environment for Linux developers and portability from exiting Linux applications. Space disk limitations in mobile devices and instabilities are some of the current problems that should be solved to create a solid line of evolution.

6.4.7 Linux-Based Devices

Linux-based mobile phones represent a promising line. Although there are some examples of devices available from 2007 it was in 2008, with more than 20 devices launched on the market, that this trend started to get market share. The earlier-mentioned Android environment is an important point for Linux mobile phones. Three handset manufacturers have implemented phones running Android software during 2008: HTC, with the HTC G1 model (Google, 2008a) sold by T-Mobile; Qigi, with the Qigi i6 model (Google, 2008b), which will be sold in the Chinese market; and Kogan Technologies, with the Kogan Agora model (Arora, 2006). Qigi will be commercialized in 2009, although most of the market parameters are defined.

Before closing this chapter, it is necessary to mention the OpenMoko manufacturer (OpenMoko, 2008a) and the 'crusade' that they are following. In 2007, this company decided to change its strategy to create an open-source project (OpenMoko, 2008b). They opened not only the software of its devices but also released the CAD files. In 2008, they announced the release of the schematics for its products. They are selling the Neo FreeRunner phone for advanced users and they have planned to sell for the mass market when the software becomes stable. The Openmoko stack, which includes a full X server, allows users and developers to transform mobile hardware platforms into unique customized products. The license itself gives this possibility. OpenMoko represents a completely different way of creating mobile handsets in a very fragmented world, where the user and his/her necessities are still out of the scope of the manufacturers.

6.5 Research and Development

Beyond the current established trends, the R&D community is working in differently termed initiatives to push forward the future of the ICT panorama of services and technologies. Some of them need to be considered when observing paths to follow in the evolution of the IMS itself or the telco infrastructure in a broader perspective. These could be specific items, like the immediate strategy to leverage on IMS-enabled terminals by exploiting the potential of mobile browsers as applications containers or the future protocols considered for a multimedia telecom infrastructure that may supersede the SIP or even the IMS as a concept, or broader topics, like the considerable next-steps that may notoriously affect the ICT business, the so-called Web 3.0 and Telco 3.0. Acting as an umbrella, the Future Internet is the embracing initiative with clear political influence to sustain the digital economy, and is also considered.

In all these initiatives, the decisive role of R&D investment will shape the landscape of ICT and, more specifically, the telco business of tomorrow where, as already mentioned

in this book several times, the IMS component might prove to be a booster from the technological side.

6.5.1 IMS Exposure to Browser

Developers who have ever tried coding up a mobile client application noticed that the huge variety of mobile operating systems makes it difficult to build rich applications that work on every device. This also applies to the case of developing applications that benefit from the IMS service capabilities. A trend in simplifying this development is to create web applications that run in every mobile device independently of its operator or manufacturer. Developers could use the same coding skills to create mobile applications with simple technologies and developing frameworks coming from the web. This approach is followed by Adobe in the AIR framework, which enables the creation of PC desktop applications in a similar manner as if they were web applications. Besides this, developing mobile applications utilizing the mobile web browser as container for the client-side application logic has benefits in terms of upgradeability of the application. Web applications are inherently easier to upgrade and modify their look-and-feel without forcing the user to download new software. Beyond this, if these mobile web applications could work offline, users would be able to use them when they are disconnected from the network.

Nevertheless, for applications that require interpersonal communications or other enabling capabilities like those provided by IMS – or more generically, the telecom network via the terminal device – there is a growing need for mechanisms to expose terminal capabilities to the web application running in the mobile web browser. Following this principle, the browser acts as a service container with the ability to provide applications with a local environment. This comes with controlled access to the terminal features, including IMS services and other terminal specifics, like the accelerometer, GPS, vibrator and camera. In addition, the cloud computing principles are applied to telecoms: local contact lists, call logs, multimedia local store, messaging local store, etc.

In this sense, nowadays there are some initiatives that might evolve towards the concept of IMS and terminal capability exposure to the browser. Among those OMTP Bondi and Google Gears Mobile are discussed in this section.

6.5.1.1 OMTP Bondi

The open mobile terminal platform (OMTP, 2008b) is a forum founded in 2004 by eight mobile network operators. The OMTP was set up to discuss standards with manufacturers of cell phones and other mobile devices with the aim of simplifying the customer experience of mobile data services and improving mobile device security. Bondi is a new initiative launched in 2008 by the OMTP. The name takes reference to an Australian beach following the corporative slogan ‘It’s safe to surf’.

The OMTP Bondi initiative tries to solve the problem that an application written for one phone must be rewritten again and again if it is to work on all phones. This is the mobile platform fragmentation problem. The cost to the mobile industry of this inefficiency is huge. This situation creates barriers that slows down time-to-market, limits market size and increases customer confusion.

OMTP seeks to remedy this problem by helping to address the way in which the existing Web 2.0 environments are moved on to mobile devices. Mobile devices offer new capabilities to web service developers that make them very desirable, but also present new security issues. OMTP is defining the key interfaces that enable the mobile web platform to access sensitive functions on the mobile device. OMTP is defining appropriate security mechanisms that will enable new services and create user trust.

By delivering high level requirements, draft specifications and a reference implementation (OMTP, 2008a), which will cement those requirements in a solid foundation, OMTP is seeking to provide a consistent, highly desirable and secure interface from web applications to billions of devices in the future. The Bondi solution lies on the following functionality:

- Application packaging. Defines the precise mechanisms by which an application can both declare its identify and define any dependencies it may have on either device capability or other software components.
- Extensible APIs. Where the functional interfaces define static mandatory APIs this work stream analyses the mechanism by which new APIs can be dynamically installed on the phone.
- Policy management. Defines the mechanisms by which policies can be changed/managed by remote entities.
- Security policy definition. A precise, flexible and interoperable definition of when each application can access a particular resource, where an applications identity is determined by its security credentials.

The standards that Bondi is seeking will allow a web page or a widget access to specific functions of a phone, like start calls, send SMS and access to the contacts within the device address book or geo-location functions. To make it possible, Bondi is also putting a lot of focus on solving the security and privacy issues. We have to avoid situations where a widget can take a photo, send uncontrolled SMS messages or start a call without customer authorization. Things like this will damage the consumer trust in these innovative functionalities.

OMTP have joined the World Wide Web Consortium (W3C) as a member. This membership of W3C will allow OMTP to make submissions directly into the key W3C activities affecting Bondi. These will, for example, include work within the W3C Web Applications Working Group helping to standardize widgets. The ability to protect customers from malicious services originating on the web is paramount, as is the consistency of the interfaces for the developer community. By working with W3C, OMTP will be able to help deliver consistency and security and assist in making web services ubiquitous across different handsets.

6.5.1.2 Google Gears Mobile

Gears, created in 2007, is free and open source software offered by Google (2009b). Gears is a plug-in that extends the browser to create a richer platform for web applications. The broader goal is to close the gap between web apps and native apps by giving the browser new capabilities. For example, webmasters can use Gears on their websites to let users access information offline or provide contents based on customer geographical location.

After the introduction of Gears for the wide web environment, Google launched the mobile version. Gears for mobile (Google, 2009a) is an open-source mobile web browser plug-in launched in 2008. It lets developers create web applications with added speed or functionality, such as offline capabilities that allow web applications to work without an active Internet connection. If users are using a mobile web application and suddenly lose your cell connection, users can access Gears-enabled mobile web applications offline. Initially, Gears for mobiles is only available for Internet Explorer Mobile on Windows Mobile 5 and 6 devices. However, Google has said that it plans to expand support to other browsers and cell phone platforms. Presumably, that includes Opera, Safari on the iPhone and its own Android software.

In practice, Gears for mobiles would allow you to write a mail by Gmail, create a document using the text editor of Google Docs or use the Google spreadsheet while the network connection is down. Thus, users could send messages or documents using a short time connection. This could be very useful to customers who are scared by the mobile Internet high prices to use these services. Nowadays, in European countries, most customers do not use their mobile phone to access Internet services because of this. Gears is useful in case of lost Internet connection, accidentally or not.

6.5.2 Future Session and Service Protocols for IMS

Is SIP really in danger of its usage in IMS? If this is so, what are the menaces? Does any protocol really exist that might make SIP obsolete? As a matter of fact, it is not possible to answer 'Yes' to the first and third questions. At least the Internet does not reveal any exact substitute for SIP in its application to telecom networks, but this does not ensure the emptiness of the list of menaces. In fact, there are a few protocols that, according to certain studies (or professional opinions), perform better than SIP at tasks generally performed by SIP or where SIP is present somehow. SIP seems to be basic for core signalling, and that is a big difficulty itself to get SIP made obsolete, but, with regards to the media exchange, there are a few competitors that may enable the shift. In this section, comparisons with other protocols are offered (proprietary protocols are considered out of the scope of this book), and pros and cons are analysed in order to reason whether those protocols could become a preferred option to substitute SIP in a certain areas.

6.5.2.1 H.323

This is an ITU-T recommendation that used to be the only protocol for the VoIP calls setup until SIP (IETF standard) appeared. SIP offered more simplicity, ease of development, processing time reduction and overhead reduction. All these contributed to why SIP has become more popular than H.323 (Levin, 2003).

However, both protocols have evolved along the years and have learnt from each other, and even more work is currently being done to extend their specifications. SIP has become more complicated and H.323 has been simplified, so their performance is located now more or less at the same level. The literature considers that H.323 may be, again, a SIP competitor.

In terms of functionality and services that can be supported, H.323 and SIP are very similar. However, supplementary services in H.323 are more rigorously defined and therefore fewer interoperability issues are expected to arise. Furthermore, H.323 has better compatibility among its different versions and better interoperability with the PSTN. The two protocols are comparable in their QoS support. In general, SIP primary advantages are its flexibility to add new features and its relative ease of implementation and debugging (Papageorgiou, 2001).

However, one key concept that might make H.323 impose upon SIP is the fact that H.323 specifies services while SIP just specifies the signalling needed for the services to be delivered. For service-oriented platforms, H.323 implementations might be more interesting.

6.5.2.2 H.325

Since 2005, ITU-T study group 16 has been working on a project that aims to widen the multimedia communications vision depicted by H.323. The project, called the 'Advanced Multimedia System (AMS) Project', relies on the new protocol H.325 to drive the development of a third generation multimedia terminal and system architectures able to support emerging, media-rich applications that fall outside the bounds of traditional call-based communication platforms. These applications include highly converged media applications involving multiple personal and public devices, enterprise systems and network services in support of communications, collaboration and entertainment (ITU-T, 2007). The main difference between H.325, on the one hand, and H.323 and SIP, on the other hand, is that H.325 is focused on applications and device enablement rather than call setup. It has been decided that the H.325 system will utilize XML for signalling.

AMS is based on the following architectural components:

- Container. This is the device that represents the user to the network (e.g. a desk phone, mobile phone or softphone application).
- Application protocol entities (APEs). These are the applications that register with the container to provide the user with voice, video and data collaboration capabilities.
- Service nodes (SNs). These are the network entities that enable the container to establish communication with a remote entity, facilitate NAT/FW traversal and provide their network-based services.
- Application servers (AS). These are elements in the network that provide various services.

The AMS architectural development process has several important functional goals:

- AMS will have a diverse application support model, such that many different types of applications can utilize the architecture. Separation of base signalling architecture and applications will ensure that application developers do not have to be concerned with the underlying signalling details.
- Security and privacy will be built into AMS and will be both policy aware so that they can be effectively managed and interdomain aware so that they can function in a diverse, global environment.
- Network boundary awareness will be built into AMS, so that AMS systems function well between networks using different addressing universes, such as NAT, IPv4/v6 and the PSTN.

- Strong versioning, tight vocabularies and efficient signal coding methods will be employed to ensure that interoperability problems and coding errors are minimized.
- Location awareness will be included as a core application functionality of AMS. This will support emergency management requirements, but will also enable presence applications and new applications to take advantage of location information.

H.325 is currently under a standardization process. An initial release is expected in 2010. Therefore, it is too early to say that H.325 will make SIP obsolete, since it is necessary for both protocols to be implemented for a proper comparison.

6.5.2.3 XMPP

The extensible messaging and presence protocol (XMPP) is a protocol for streaming extensible markup language (XML) elements in order to exchange structured information in close to real-time between any two network endpoints. While XMPP provides a generalized, extensible framework for exchanging XML data, it is used mainly for the purpose of building instant messaging and presence applications (Saint-Andre, 2004). Although SIP can carry XML bodies, it is not optimized for messaging and presence applications (its main purpose is the session establishment). For these applications, there is an SIP extension: SIP for instant messaging and presence leveraging extensions (SIMPLE).

SIMPLE may interoperate with XMPP but, unlike XMPP, it is not optimized for the advanced exchange of structured data and provides only a subset of messaging and presence functionality. In this sense, it might be stated that XMPP is an SIP substitute for messaging and presence applications.

Further on, if we try to see whether XMPP can stand in for SIP at more scenarios, we find 'Jingle', which is an XMPP extension for the transport of RTP traffic (another typical use of SIP). Jingle also supports the SIP-like capabilities of early media and session (re-)negotiation, but does not support as many codecs as SIP. Besides the applications variety, XMPP does not imply big implementation costs (thanks to the Jabber open-source community) and there already exist several IETF standard specifications (RFC 3920, RFC 3921, RFC 3922, RFC 3923). The question is: Is all this enough to say that XMPP might make SIP obsolete? A deeper study is needed, but it really looks like a well-positioned candidate.

6.5.2.4 IAX

The inter-Asterisk exchange protocol is an application-layer control and media protocol for creating, modifying and terminating multimedia sessions over Internet protocol (IP) networks. IAX was developed by the open-source community for the Asterisk PBX and is targeted primarily at voice over Internet protocol (VoIP) call control, but it can be used with streaming video or any other type of multimedia (Capouch, 2007). IAX is an 'all in one' protocol for handling multimedia in IP networks. It combines both control and media services in the same protocol. In addition, IAX uses a single UDP data stream on a static port, greatly simplifying the network address translation (NAT) gateway traversal, which is the main advantage of IAX over SIP. IAX employs a compact encoding which decreases bandwidth usage and is well suited for the Internet telephony service. In addition, its open

nature permits the additions of new payload types needed to support additional services. Since IAX is dedicated to Asterisk devices, it is not a real competitor against SIP.

6.5.3 Web 3.0: from the Social Revolution to a More Usable Internet

As mentioned in earlier chapters of the book, the main change produced by Web 2.0 is the social revolution that happens in the Internet. The user has become an active actor, 'the leading man/woman', controlling and deciding over the contents and applications placed on the Internet and actively placing contents. Web 2.0 technologies were born at the end of the 1990s and the beginning of 2000s, but were successful some years later. When attempting to talk about a potential Web 3.0, the main feature would be how to include more usability in the Internet and, at the end, make it an easier world for the end-user. Figure 6.12 shows how the Internet will evolve to 2020 (Spivack, 2008).

There is no clear definition of the Web 3.0 term yet, but most researchers talk about emerging technologies such as Semantic Web, enhanced broadband technologies such as FTTH (fiber to the home) or LTE (long-term evolution), modular web applications and enhanced computer graphics to massive 3D deployments on the Internet. Thus, Web 3.0 represents a new era in the web, where the web structure is enriched thanks to an enhancement of collaboration between people, search quality, advertisements, etc. The new web will represent more usability in the search, from the keyword search where users and service providers need to have common patterns to produce good search results to natural language or semantic search enhancing the performance of the search.

A common term under Web 3.0 is Semantic Web. What is expected under this term is to create a vision of information understandable by computers. Today's computers need

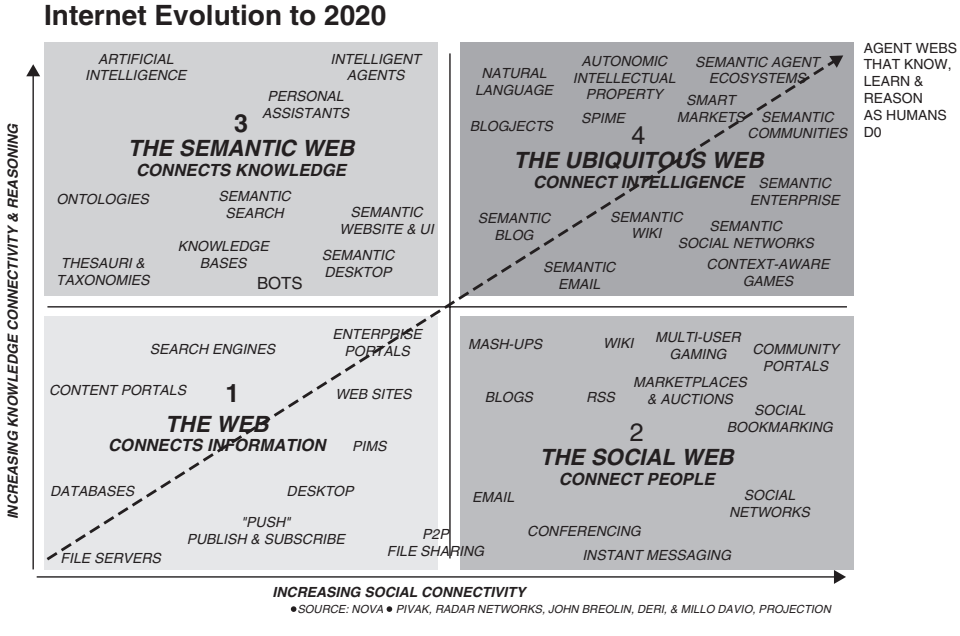

Figure 6.12 Internet evolution.

human guidance because web pages are designed to be read by people, not computers. With Semantic Web, computers could perform most of the work around finding, sharing and combining information on the web. As Tim Berners-Lee coined (Berners-Lee, 1999):

> *I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize.*

A semantic approach to the web has some advantages, such as more accurate searches, less computing resources required, working using both structured and nonstructured data and allowing more intelligent applications to be created that share data between them. However, today there are not many tools using this approach; scalability is a current limitation and there is a challenge concerning who is creating the required metadata. Semantic Web is considered as an open database over the current web. Some key issues are:

- Intelligence is inside data and out of the applications.
- The meaning of data and the way to use it is part of the data (auto-descriptions) enabling applications to be more intelligent thanks to the data analysed.
- Data can be shared and linked more easily. The Semantic Web is composed of open standards to create this new web.
- RDF (resource description framework). This stores data like triples (a subject–predicate–object expression).
- OWL (ontology web language). This defines systems of concepts called 'ontologies'.
- SPARQL (SPARQL protocol and query language), for queries in RDF.
- SWRL (Semantic Web rule language), for rules definition.
- GRDDL (gleaning resource descriptions from dialects of languages). This transforms data to RDF.

There are more possibilities than RDF/OWL to express semantics in the web, like micro-formats, chain of tags, controlled taxonomies and vocabularies, etc.

After seeing many changes in the Internet and related technologies, it seems that users, except those digital natives, spend more time that expected to accept new functionalities. In fact, one common characteristic of companies is that most of them still have not yet adapted their organizations to the Web 2.0 trends and business models with the goal to increase sales or to maintain the current position in sales, as stated by Antonio Fumero from Win Win Consultores (Consultores, 2008). As defined by experts like Nova Spivack (2008) in trends about the Internet, Web 3.0 may have success during 2010–2020. Therefore, it is perhaps still too soon to define with accuracy when the new version of the web will come, but what is quite clear is that the promised value is important for the end-user.

On this front, users are demanding more and more usability on the web. Usability needs to join the simplicity, accuracy, intelligibility and the ability to reduce tedious activities by the user. Whether Web 3.0 will bring those capabilities to the user is still to be demonstrated, but what has happened with Web 2.0 is an irreversible social change. More and more users are joining the Web 2.0 effect, of course without knowing about the underlying technologies.

If technologists have learned the lesson of communicating clearly the value to the users, the new web will be adopted quickly. If not, then Web 3.0 will take some more years to be accepted by the user, or, what is worse, could be deprecated by users. An underlying question is: If people are happy about more and more activities being controlled by computers, are we building the Matrix?

6.5.4 Telco 3.0

Telco 2.0 is known as the paradigm or industry model where telecom companies try and get benefits from sources that are not their own. The business model is impacted by the transformation in relationships with both users and competitors (Telco 2.0, 2006):

- Users evolve from simple consumers to 'prosumers' (producers + consumers). Telco service providers expose a series of services and enablers that bring together numerous applications and types of content from a variety of sources to enable people to produce and consume composite services.
- Competitors change into 'coopetitors': partnership and cooperation are invesiged in order to achieve service synergies. Telco 2.0 embraces the principles of Web 2.0, which shows that open interfaces are to be used. This means that anyone (user or company) can create and upload applications and services. With this, telco operators start to behave as Internet content providers. Keeping all this in mind, Telco 3.0 can be regarded as the addition of Telco 2.0 plus a series of features that can be grouped into two big fields (Canal, 2007):
  - Enabling infrastructure. More flexible, more effective, less expensive Infrastructures tend to be self-sustained.
  - Innovative technologies. These involve P2P, autonomics, semantics (ontologies/folksonomies), etc. Like Telco 2.0, Telco 3.0 embraces the principles of Web 3.0, which is regarded as an evolution of Web 2.0, where machines can gather, elaborate and combine data and information semantically, much as humans browse the web today, as described in the previous section.

A WebTelco 3.0 vision is elaborating the model of an open, distributed and flexible service ecosystem built as an ecology of components/agents learning and self-adapting to prosumers' behaviours. In particular, data, services and telco-ICT enablers are, in the same way, abstracted by lightweight components/agents capable of self-organizing semantically to serve (in an adaptive and goal-oriented way) prosumers' needs (Cascadas, 2009). From the business point of view, together with the 'coopetition', the concept of 'economy of abundance' arises; by offering a lot of services for free, telco service providers may gain and develop new sources of income.

Telco 3.0 does not count with a closed set of defining characteristics so far. Telco 3.0 is a wide, vague enough concept, so that it becomes extended with any single new research project that is started in the area. The main lines of investigation that support Telco 3.0 at the moment are the following:

- Future Internet services: Web 3.0 based, user-centric ecosystem based, a search for maximizing user experience and satisfaction. Open APIs are available for 'prosumers'.

- Distributed service eco-systems based on autonomic agents or components and peer-to-peer technologies and solutions – the situated and autonomic communications (SAC) paradigm.
- Situated communications (Sestini, 2004): reacting locally on context changes, ranging from sensor networks to virtual networks of humans and considering technological, social or economic strategic needs.
- Autonomic communications (Sestini, 2004): network elements autonomously interrelated and controlled, learning the desired behaviour, with self-organizing and radically distributed features.
- Service platform virtualization.
- Security, resilience.
- Self-organization, evolution, management, healing, synchronization, etc.
- Situation awareness: self-ware, knowledge-ware, sensor-ware, service-ware.
- Ambient intelligence: ubiquitous computing, ubiquitous communication and intelligent user interfaces.
- Pervasive computing and communications.
- Creation of distributed and dynamic ontologies and folksonomies.
- Technologies enabling collaborative services with direct interactions of prosumers.
- Interaction of new paradigms with society: social networks, user-generated contents and services.

Telco 3.0 seeks to find the following industrial benefits:

- Optimization of Telco-ICT service frameworks (Capex savings). Since all services will be developed within a common framework, capital investment is needed only for building this framework.
- Simplification of management (Opex savings) of Telco-ICT service frameworks. Since all services will be developed within a common framework, common management for all services/enablers may be implemented, yielding operational cost reductions.
- Introduction of innovative SAC services. The SAC paradigm (described above) contributes to the optimization of the service framework and the simplification of management. The 'situated' feature intends to improve the user experience; thus SAC services are more attractive to clients who aim for service personalization and environment customization. The 'autonomic' feature enhances and simplifies fault handling and resilience, as well as making possible cooperation with partners and open 'prosumption', which are new business opportunities after all.
- Research in business processes modelling for workflow enhancement. Improving processes efficiency avoids waste, reduces time-to-market and, in general, increases performance. Research in business processes modelling is a long-term activity since, due to technological innovations, an optimal process one year may be obsolete and poor performing the following year. Business processes modelling is a general worry in companies nowadays, and a number of organizations are leading research activities in this sense. For instance, the TeleManagement Forum (TMF, n.d.) is developing standards for telco business processes.
- Partnership synergy. Since Telco 3.0 is 'coopetition' oriented, sharing the service framework with partners will yield enhanced, composite services that will be ready to use within shorter terms.

- New sources of income yielded from 'economy of abundance'. The typical example that illustrates the concept of 'economy of abundance' is its appliance to music. Illegal downloads are provoking the condition that selling records is less and less a good deal. Applying 'economy of abundance', all music should be downloaded for free and revenues will come in the form of concert tickets (since the music is delivered for free, it can reach more people, including those who would not have bought an album in the first place; as a result, more people become fans), merchandizing, clicks on the adverts on the webpage where you can download the music, etc.
- Revenues from services not made by the Telco company. Users and other companies will develop a number of services. The Telco company may charge a small fee for development toolkits and make business from developed services in a direct way (selling the service) or indirect way (like advertisement add-ins, for instance).
- Benefits from the 'Long Tail' model. The theory that the Internet enables businesses to sell a large selection of goods in small quantities to a large population of customers at minimum cost was popularized as the 'Long Tail' by Chris Anderson in a wired magazine article in 2004 (Anderson, 2009).
- Future Internet development. Web 3.0 is the next step: what will come afterwards? What will be Web 4.0? What will be the business opportunities? The only way to find out is by taking the next step forward, joining the WebTelco 3.0 paradigm.
- Churn reduction. This is a logical consequence of clients' satisfaction, motivated by the availability of better services at a lower cost.

6.5.5 Future Internet

With over a billion users worldwide, the current Internet is a great success – a global integrated communications infrastructure and service platform underpinning the fabric of the world economy and society in general. However, today's Internet was designed in the 1970s for purposes that bear little resemblance to current and foreseen usage scenarios. The effect of that is summarized in the following points (*et al*., 2008b):

- The Internet has grown from an experimental research network to an infrastructure supporting the economy as well as the provision of societal services.
- The Internet is becoming pervasive and ubiquitous, with already 25% of the world population having access to it.
- It has enabled user and consumer empowerment, through the emergence of e-Commerce and social networks.
- It has helped the modernization of public administration through the emergence of e-Government, e-Health, e-Education. Internet use is also expected to contribute significantly to solve emerging challenges such as climate change and energy efficiency.

The Organization for Economic Co-operation and Development (OECD, 2008) has politically acknowledged the importance of the Internet in the present days and in the future for the economic, cultural and social development of modern societies. OECD's Seoul declaration (OECD, 2008) for the future of the Internet economy stresses the achieved commitment to promote policies to facilitate the expansion of Internet access and use worldwide; the promotion of an Internet-based innovation, competition and user choice; to secure critical information infrastructures and ensure the protection of personal information intellectual

property rights; and to create a market-friendly environment for convergence that encourages infrastructure investment, higher levels of connectivity and innovative services and applications.

Mismatches between original design goals and current utilization are now beginning to hamper the Internet's potential. A large number of challenges in the realms of technology, business, society and governance have to be overcome if the future development of the Internet is to sustain the networked society of tomorrow. Thus, multiple initiatives in the scope of the future Internet have been launched in different regions worldwide with the objective of sustaining the competitiveness of the regions concerned. In the US, the most well-known initiatives are GENI and NetSe initiatives of the US National Science Foundation (www.geni.net). In Japan, the AKARI initiative supports a Japanese new generation network (NWGN) project. For these two initiatives, clean slate approaches are considered with a time-to-market horizon in the 2015 to 2020 range.

Perspectives in Europe forged the emergence of the Future Internet Assembly (FIA, 2009), kicked off on the occasion of the Bled Conference of March 2008. As the Assembly narrates, the FIA gathered sector actors in the research space to define a European approach towards the future Internet, also leveraging the achievements of many national research initiatives. The FIA also addresses nonresearch issues, in view of easing the future deployment of the technologies and systems, such as standardization and international cooperation.

The main goal of the FIA is to confront the various technological and architectural approaches deriving from the multiplicity of usage scenarios and requirements applying to a future Internet. It is expected that this will help the emergence of cross-sector consensus and pave the way towards the later emergence of common architectural approaches and standards, since there is no agreed definition of what a future Internet should cover, both in terms of research and nonresearch issues. Early research work on the future Internet has primarily concentrated on networking issues and on the need to support a range of innovative applications with enhanced management capabilities and a particular focus on security issues. On the other hand, it is more and more widely accepted that future Internet issues are driven by a very large set of new application requirements. It is hence nowadays commonly admitted that the future Internet needs to be tackled from a holistic perspective, beyond the 'network' perspective but with a broader 'technology focus', which is mostly related to industrial (ICT) competitiveness. In the case of Europe competences and leadership in ICT, this includes notably technical areas such as mobility, very high rate end-to-end broadband, security and even 3D media. It is, however, also clear that a 'technological only' focus would not be the right approach, and therefore a need emerges to take into account the applications dimension very early in the process, eventually leading to future Internet technologies. With respect to applications, a key feature of many. Future Internet initiatives relates to the provision of testbeds and experimental facilities, which foster opportunities to make novel classes of innovative applications available and can have huge economic and social impacts outside the ICT sector. A typical example would be sensor-based applications for remote health care and environmental or energy efficiency control. Experimental research facilities offer a good opportunity not only to evaluate the viability of the technical solutions but also to consider more effectively and dynamically the application requirements as part of the research process. User-driven innovation is supported, by enabling user access to the experimental research facilities. It is more and more

recognized that the availability and mastering of complex experimental research facilities and testbeds provide a competitive advantage in terms of effectiveness of R&D and innovation processes.

The need to make the future network and service infrastructure trustworthy is another key aspect to be addressed in designing the future Internet. Trust, security and privacy protection deserve specific attention as they have to be addressed horizontally, across all possible usage and technological scenarios. With these considerations in the landscape of the European vision, the following high level objectives are considered to drive research pursuing a future Internet:

1. To accommodate unanticipated user expectations together with continuous user empowerment.
2. To become the common and global information exchange environment of human knowledge.
3. To lever and evolve information and communication technologies, capabilities and services to fulfil increased quantity and quality of Internet use.
4. To support open cultural, scientific and technological exchanges across all regions and cultures, as well as within single communities.
5. To be ubiquitously accessible and open (at physical, connectivity and information levels).
6. To be secure, accountable and reliable without impeding user privacy, dignity and self-arbitration.
7. To support mobility, have widespread ubiquitous availability and be capable of assisting society in emergency situations.
8. To support effective and efficient performance management features based on context, content, etc.
9. To support innovative business models that allow for all entities equal access to the services and service provision markets.
10. To be carbon neutral, energy efficient and environmentally sustainable.

In order to cope with these high level objectives, finer challenges are depicted in the future landscape for the Internet to address a multiplicity of new usage patterns and the plethora of service requirements not foreseen when the Internet was designed. Even if there is no agreed definition or standard describing the nature of the future Internet, consensus on most of these challenges is driving the research. These challenges, described below, can serve as a set of technical features that may describe what the future Internet would resemble:

- Mobility, to allow a native support of moving scenarios and an open Internet fully available on the move.
- End-to-end very high throughput, to support data-intensive usage scenarios, such as medical imaging transfer and processing.
- Security and trust, involving processing of sensitive personal data or financial transactions.
- Internet of things to support device connectivity (RFID, sensors) and coupling of virtual world data with physical world information in a scalable and efficient way.

- Internet of services: service architecture enabling dynamic, secure and trusted service compositions and mash-ups while allowing the user a proactive role (user-generated services).
- 3D and media intensive support: 3D virtual environment, possibly coupled with physical world information, beyond games.
- Negotiated management and control of resources: dynamic and predictive network management, infrastructure observability and controllability to support variability of the business model, from a best-effort low level of control towards full real-time management of quality of service, security level, etc.
- User-controlled infrastructure: ad hoc network and service composition, user-driven deployment scenario and control of connectivity.

Beyond research, it is considered that important nontechnological issues have also to be covered. International cooperation with those regions having initiated large activities is seen positively, especially if it provides opportunities to run global experiments through interconnected research facilities. International cooperation is also deemed necessary for what concerns the standardization effort as well as interoperability testing that will have to be undertaken once novel technological approaches are identified at the EU level.

Finally, the future Internet should take due account of emerging societal challenges, in addition to more classical issues such as user acceptability and the need to increase further the role of the Internet as an enabler of innovation ecosystems. Such additional issues include green computing and support for the energy efficiency of multiple user sectors.

6.6 Concluding Remarks

The chapter has attempted to provide a possible snapshot of the future. The authors understand, however, that this is a very fragile and unpredictable domain with numerous established players and also new players who may influence the future. With that in mind, we believe that the book, in general, sheds new light on the prospects of IMS in a way that has never been done before. It also gives the most comprehensive evaluation of current standing and future possible positioning of IMS in the service-oriented landscape. In its final part, the book also provides a unique vision of the road ahead. We hope that it will help researchers as well as industry players and the strategy makers in their efforts to move forward in the currently turbulent sector.