Author(s): Licciardi, Carlo Alberto; Falcarin, Paolo.
Article title: Technologies and Guidelines for Service Creation in NGN
Year of publication: 2003
Citation: Licciardi, C.A. & Falcarin, P. (2003) "Technologies and Guidelines for Service Creation in NGN." In: Proceedings of the 8th ITU International Conference on Intelligence in Networks (ICIN 2003), Bordeaux, France, April 2003.

Technologies and Guidelines for Service Creation in NGN

Carlo Alberto Licciardi[1], Paolo Falcarin[2]
[1]*Telecom Italia Lab* via Reiss Romoli, 274 Torino 10148, Italy
[2]*Politecnico di Torino, DAUIN* corso Duca degli Abruzzi, 24 Torino 10129, Italy
e-mail: carlo.licciardi@tilab.com; PaoloFalcarin@polito.it

ABSTRACT

Network Operators can see next Generation Networks (NGN) as new revenue stream, thanks to the potential they could have in increasing the service offering. Therefore it's important to understand how proposed technologies and solutions in NGN market can enable, flexible and easy service creation [4] . This paper presents the result of the investigation of Eurescom P1109 project [1] in the area of advanced technologies that enable the introduction of new services in NGNs [6]. These technologies are evaluated with respect to some key evaluation criteria and then a comparison is provided.

1 Introduction

NGNs have been promoted to network operators as a way to decrease operational costs of existing infrastructure. Actually there is no clear business analysis that has proven this thesis. On the other hand NGN can be seen by network operators and service providers as a new revenue stream from their potential to increase service offerings. Therefore it is of paramount importance to understand how proposed solutions in NGN market can enable flexible and easy service creation both to service providers and 3rd party application developers.

EURESCOM P1109 Project "Next Generation Networks: the Service offering standpoint" [1] has addressed this issue by evaluating NGNs service platforms in terms of functionality, programmability, flexibility, openness, and inter-operability. In other words the objective has been to put to the test some of the major benefits promised by NGN, namely productivity, creativity and new revenues from new business opportunities, and to see how well current product offerings supported these capabilities, in terms of available tools for NGN service development; evaluating how much easy and efficient is to develop and deploy NGN services [6]; evaluate product maturity, standard compliance and interoperability. Among these issues this paper focuses on an analysis of different service creation technologies, in order to show which options are available to developers.

These technologies are evaluated with respect to some key evaluation criteria (programmability, usability, network capabilities, kind of interfaces) and then a comparison is provided by means of a sum-up stable, followed by some useful guidelines for network operators that want to migrate to NGN in a profitable way.

2 Assessment of service creation technologies

In this section we describe the evaluation criteria and the comparison of some of the more interesting technologies that can be used for service creation in NGN with respect to the identified criteria (for a detailed analysis refer to [2].

2.1 Evaluation criteria

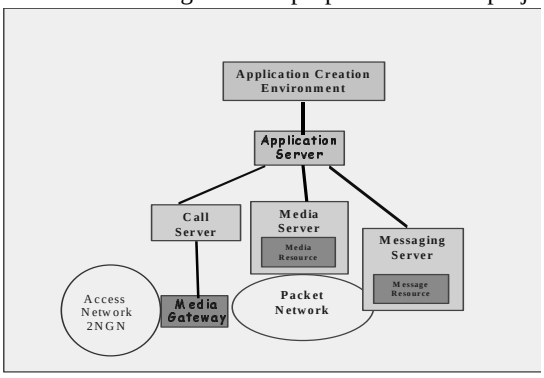
In this section there is a definition of evaluation criteria used for classification and comparison of different service creation technologies, in short: supported network capabilities, mapping towards reference architecture, interface abstraction, kind of interface (and description language), suitability for 3rd party development, easiness to use, industry support, maturity, and future-proofness.

The first criterion is based on *Network capabilities*, i.e. the abstraction of underlying network infrastructure that can be used by application developers to exploit network functionalities; they can represent both functional (e.g. call control) and non-functional (e.g. authentication, logging…) aspects. Parlay/OSA [7] consortia have defined a set of capabilities, which have been considered as a basis for the following definitions of different network capabilities:

- **Non functional**: Framework Functions is a part of the Open Service Access (OSA) API interface which provides management capabilities needed for accessing service interfaces in a secure and manageable fashion. It controls authenticated access to Service Capability Servers (SCSs) and also supports standard interfaces like service registration, service discovery, authentication, etc.
- **Functional aspects**: the following Service Capability server have been defined:
    - Generic Call Control (GCC)
    - Multi-Party Call Control (MPCC)
    - Multi-media call control (MMCC)
    - Conferencing Call control (CCC)
    - 3rd Party Call Control (TPCC)
    - Generic User Interaction (GUIN)
    - User location (UL)

- User status (US)
- Data access session control (DASC)
- Messaging
- Terminal capabilities (TC)
- User profile (UP)
- Matching to CAMEL/IN

The second criterion defines which place a technology covers in the categorization proposed in P1109 project



as reference architecture, depicted in figure 1.

Figure 1. The P1109 Reference architecture

This layered architecture defines a distinction among technologies, depending on their characteristics: application server layer includes technologies used to execute services, programmed with tools, represented by the Application Creation Environment layer; call server layer includes technologies handling routing and delivery of voice calls; media server layer represents technologies involved in multimedia communications, and messaging server stands for entities handling messaging and asynchronous communications. Media Gateway layer represents networks related technologies.

Third criterion is the evaluation of interfaces offered by technologies to developers; the interface evaluation defines the level of abstraction (AIL), the kind of interface (KOI), and its type of Interface Definition Language (IDL).

Regarding the abstraction level of an interface, an abstract interface hides technical details of the underlying technology to the developer, in order to gain more portability, easiness of use, concise programming; a mid level interface hides parts of the details of the underlying technology, but still requires some level of knowledge form developer, and also the ability to choose controlling low level details; a low level interface provides detailed access to the underlying technology (e.g. a network protocol stack), such that the application developer has to manage with less portable and more lengthy code, using technology specific API that are more difficult to learn.

The "kind of interface" should describe the communications method by which the technology in question is exposing network capability to external systems. This should include the following categories:

- Application Programming Interface: that can be Local, when the API is only resident on the local execution platform or Distributed, when it is accessible from distributed nodes in the network.
- Protocol based interface: if it is a direct interface to a protocol stack
- Scripting Language: if the information used to program a technology is passed using scripting languages 'interpreted' at runtime (e.g. XML-based and policy languages).

The type of interface defines the language used to define its API; we can classify them in Computing language based (Java, C++), middleware based (OMG IDL in CORBA, WSDL in Web Services), or data definition based (e.g. XML DTD for CPL or XML Schema).

Another criterion used in evaluation is programmability: that is suitability to 3rd party application development (TPAD), which describes the qualification of the technology in support of application development by 3rd party developers, and the suitability to 3rd party service provider (TPSP), which should describe the qualification of the technology in support of 3rd party service provider hosting of applications and services.

An important criterion is also Usability or Ease-of-use (EOU): this can be measured depending on:

- The background needed by the developer, i.e. how much knowledge/experience is required of the underlying technology
- Time-to-service, i.e. how quickly it is to develop and deploy applications using this technology
- Power: the scope of what may be accomplished by using the technology in question.

An important issue to be evaluated is also the industry and standard support (IS/SS), which measures technology's availability and maturity, showing how well this technology is supported in the industry and provide a general statement as to the level of its maturity in relation to approved standards.

Finally, the evaluation criterion of Roadmap technology (RT) should identify future publicly available plans for the technology, while the Future-proofness (FP) should describe how well a technology relates to emerging technologies in the industry and possible factors that promise a future for it.

2.2 OSA/Parlay

The Open Service Access (OSA)/Parlay [7] defines an architecture that enables the inter-working between the IT applications and the telecommunications features in the mobile network through an open standardized interface, i.e. the OSA/Parlay API's. The network functionality is described as Service Capability Features (SCFs) and applications could be deployed in a third party administrative domain. The goal of OSA/Parlay is to identify and specify a Programming Network Interface in order to easily create applications using the network services provided by the Telco networks. The set of SCFs could be incrementally extended, because one of the aims of OSA/Parlay is to provide an extendible and scalable interface that allows for inclusion of new functionality in the network in future releases with a minimum impact on the applications using the OSA/Parlay interface. One of the main requirements of OSA/Parlay is to hide the complexity of the network, its protocols and specific implementation from the applications.

OSA/Parlay APIs are specified in UML. Mapping on CORBA/IDL is already available, while mapping on Web Services technology is under definition (Parlay-X). OSA/Parlay APIs are suitable to 3rd party application development, but developers need a certain level of telecommunication expertise. OSA/Parlay Framework APIs provide a secure, controlled access to network capability provided by a network operator to 3rd party service providers. OSA/Parlay APIs expose almost all the network capabilities provided by the corresponding network protocols and it eases the development of services combining several service capabilities and integrating IT applications.

2.3 Web Services

The main goal of Web Services architecture is the realization of an interoperable network of services focused on service reuse and it is suitable both to interact with 3rd party applications and to export services by a network operator or a service provider.
The Web Services can be used to export network services by exposing its WSDL (Web Services Definition Language) [17] interfaces; these services communicate using SOAP [18] (Simple Object Access Protocol), a protocol used to transport data between web services; service discovery and service registration are implemented accessing to the UDDI (Universal Discovery, Description and Integration) registry [16]; XML is used as data format for SOAP messages that rely on existing internet protocols like HTTP. Web Services implementations need that the language-dependent API must be translated in WSDL and the application server where web-services are deployed must translate incoming SOAP messages to the underlying interfaces (Java [14], CORBA...).
Different Web Services toolkits are available and some Application Creation Environments include them or offers a plug-in to handle Web Services. Toolkits can be used to translate in WSDL the existing applications' interfaces made with different languages. These toolkits also generates SOAP proxies used within the application server in order to translate SOAP messages in the underlying application language.

2.4 SIP servlets

In this section The SIP Servlet API [9] is a Java API based on the previously existing Servlet API. SIP Servlets are also a programming model where the Servlets (the applications) are hosted by an infrastructure known as a Servlet container. The SIP Servlet specification has also the objective of standardizing the following aspects of a Servlet container: the rule based mapping between Servlets and SIP requests, the security model, the servlet deployment descriptor (as an XML DTD), a jar-based file format (similar to the WAR file format used by HTTP Servlets) for servlet deployment.
The SIP Servlet API allows application to initiate and to answer SIP requests. Therefore it simply exposes SIP capabilities (both User Agent and Proxy capabilities) to the application while hiding a few protocol details handled transparently by the SIP Servlet container.
SIP Servlet API is suitable for third party service development. It could be noted that third party service development is rather simple since they are seen as Java libraries.

2.5 JAIN SIP Lite

The JAIN SIP Lite API [13] is a Java API and it is only aimed at SIP User Agent type applications that clearly define the kind of network capability exposed. Its methods expose SIP User Agent capabilities while hiding a few protocol details. The network type addressed by the JAIN SIP Lite API is very similar to the one addressed by SIP Servlets; the main differences are that: JAIN SIP Lite API doesn't necessarily address application development within an application server and it doesn't mandate a SIP proxy function within its supporting platform. JAIN SIP Lite API is standard and then suitable for third party service development.

2.6 VoiceXML (Voice Extensible Mark-up Language)

VoiceXML [15] has been defined as a technology that allows a user to interact with the Internet through voice-recognition technology. Using VoiceXML, the user interacts with voice browser by listening to audio output that is either pre-recorded or computer-synthesized and submitting audio input through the user's natural speaking voice or through a keypad, such as a telephone. VoiceXML can also be described as a

phone markup language that can be used for voice applications that provide phone access to content and information. VoiceXML is a high-level abstraction language and this means that developers with little training can use it. VoiceXML makes it easy to rapidly create new applications and shields developers from low level programming issues. VoiceXML also executes logic: main components of a VoiceXML-based speech service include tags, forms and rules that define the content and a speech browser for interpreting and presenting audio content. VoiceXML platforms are widely available and vendors are collected by the consortium VoiceXML Forum.

2.7 CCXML (Call-Control extensible Mark-up Language)

CCXML [12] has been designed to complement and integrate with a VoiceXML system, because it cannot support some needed features. For example, support for multi-party conferencing, plus more advanced conference and audio control, the ability to give each active call leg its own dedicated VoiceXML interpreter. VoiceXML needs a more effective way of handling telephony resources and for richer and more asynchronous events. For example CCXML could be integrated with a more traditional IVR system and VoiceXML could be integrated with some other call control system.

2.8 SCML (Service Creation Mark-up language)

SCML [8] is an XML-based scripting language useful to define services in NGNs. The following figure describes the relationship between SCML language and JAIN/Parlay reference architecture. The interface abstraction can be considered high level API. It is based on JCC API standardised by JAIN and therefore it's truly protocol independent. It hides network complexity and it allows handling basic events to process a call. SCML is defined using and XML schema that allow programmer to define data types as well to restrict, redefine, extend them in a similar way to inheritance in object orientation. The interface could be easily mapped onto IDL and Java. SCML is using XML schema and this means that programmers do not have to learn a new notation. Moreover it could rely on security mechanisms provided by Parlay/OSA framework. SCML looks quite easy to use like CPL [10], but it is more powerful and flexible. For example SCML scripts execution can be triggered by any event and not only by network related events as it occurs in CPL. Services such as click to dial or wake up call can be easy developed in SCML but not in CPL.

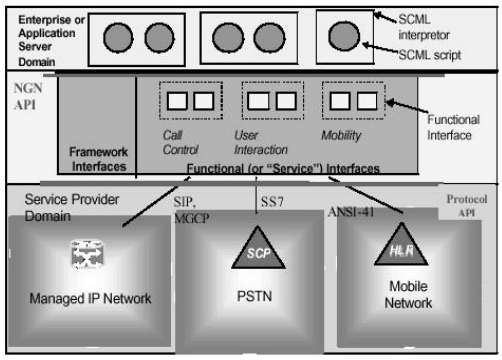


Figure 1: SCML aerchitecture

2.9 XTML (eXtensible Telephony Markup Language)

XTML [20] is an XML-based scripting language, which has been designed to provide a framework for telephony or multimedia application development without relying on a specific signaling protocol. It doesn't mean however that the application is independent of the signaling protocol, but merely that this technology is. In particular, an application can be very protocol-dependent if the support of the signaling protocol is offered at a low level. XTML is event-based: a XTML application is composed of a set of event handler, which responds to some given events. Events can be either protocol-independent (a timer expires, a session is started) or protocol-dependent (a SIP or MGCP message have been received). An event-handler is made of a set of actions, which are linked together to reflect the application call-flow, designed with a graphical Service Creation Environment (SCE). This is a proprietary tool like the application server used to interpret XTML files generated with the SCE. An XTML application is responsible for handling all of the SIP messages received (which are related to the current session), and to fully specify the SIP messages to send. However, it is the responsibility of the SIP stack to handle and easy to use but it still needs a deep knowledge of protocols specifications, in order to maintain standard compliance.

3 Overall assessment of evaluated technologies

In the following table in Figure 2, we summarize different technologies, putting in evidence their evaluated features: network capabilities offered, kind of interface and supported languages, programmability, usability.

	Network Capabilities	Interface & Language			Programmability		Usability
		Abstraction Level of Interface	Kind Of Interface	Interface Description Language	Applic. develop.	Service prov.	
OSA/Parlay	*Framework,* *CC (also MP, MM)* *UI* *UL/US* *DASC* *Messaging, (others)* *Good matching CAMEL/IN*	*Low level*	*C++ & Java*	*IDL, WSDL*	*yes*	*yes*	*No (unless a SCE is provided)*
Web-Services	*N/A* *Application-to Application middleware*	*abstract*	*XML Distributed*	*WSDL*	*Yes*	*Yes*	*Yes (with Toolkits)*
SIP Servlet	*CC* *IM & Presence* *Not matching IN*	*Low level*	*Java*	*N.R.*	*yes*	*no*	*Yes (with good SIP knowledge)*
JAIN-SIP Lite	*CC* *IM & Presence* *Not matching IN*	*Low level*	*Java*	*N.R.*	*yes*	*no*	*Yes (with good SIP knowledge)*
XTML	*No network capabilities (network capabilities are PAC dependant)* *Not matching IN*	*PAC dependant*	*XTML*	*N.R.*	*Yes*	*No*	*Yes with SCE / No without SCE*
VoiceXML	*GUIN*	*High Level*	*XML*	*DTD*	*Yes*	*Yes*	*Yes*
SCML	*CC (MCC)* *UI* *TPCC* *NO Camel/IN matching*	*High Level*	*XML (Java)*	*XML Schema*	*Yes*	*Yes*	*Yes (if SCE available) otherwise no*
CCXML	*CC* *yes*	*High Level*	*XML script*	*DTD*	*yes*	*yes*	*Yes*

4 NGN Service Creation Guidelines

This section summarises the main results of the experimentation work of the project related to the assessment of the Service Creation Process in Next Generation Networks. During the P1109 project, product selection and evaluation has shown that SIP [5] is the preferred technology to address NGN communications. The most of service creation environments (SCE) are designed on top of SIP based application servers. There are however, still several major issues that SIP products must support before they may be considered mature enough for scalable, multi-service, managed communications networks. Functions in support of service selection, QoS, billing and security are four such areas of required attention. An important step forward achieved with SIP application servers is the integration between

communication and Internet technologies. This has major implications for enabling the creation of many new innovative services for NGN networks. This evaluation has shown that as well as application servers, media servers are also a core component of NGN architecture. When compared to current PSTN networks, Next Generation Networks will be enriched by much more powerful terminals enabling the provision of new and innovative services. This remark may mean that massively used simple services with simple billing policies (e.g. flat rate) will demand much less resources from the network/application providers than PSTN services.

Application development in a NGN context is in many aspects very close to Internet application development. As a matter of fact, the main development skills required from NGN application developer are related to Java and XML. Thus NGN applications development will be accessible to a broader developer community, because it is more easy, productive and creative.

The easiness is due to the fact that need for knowledge that is specific to telecommunications is less than before and it demands for a rapid learning curve.

Productivity depends on the fact that most products don't provide a specific SCE: this allows using standard IDEs. This fact frees developers to choose the tools they are used to. Some systems provide several levels of APIs (abstract, medium and low level): this gives to the developers the flexibility of choosing the most appropriate level of abstraction for a given application (low level to control all protocol and network specific details, high level to hide network specificity). All these observations contribute for the developer productivity and, in average, a shorter time is needed for application development.

Creativity can increase because there is a move to use high-level application environments that can be used across different vendors. Having such modules can make the work of developers easier as they can concentrate in the programming aspects rather than the underlying technologies. On the other hand, the use of IT technologies makes the range of programmable features available to the developer quite wide, promoting the mix of IT functionalities (e.g.: email, instant messaging, presence, directories, web data) and telecommunications functionalities (e.g.: telephony, speech processing, quality of service, billing).

Service creation approaches in NGN can be therefore summarized in three categories: based on programmable APIs, scripting languages, or graphical SCE.

5 Conclusions

In conclusion the experiences of the project in service development phase has concluded that in general most vendors are adopting industry standard tools such as Java, XML, CPL and SIP servlets and in many cases in combination with SIP for their NGN products. SIP application servers have matured as initial product offerings and are certainly capable of small-scale deployment scenarios today. However product maturity, system stability and generally all-around management capability might still be an issue. Functions in support of service selection, QoS, billing and security are four such important areas of required attention and further investigation. An important step forward achieved with SIP application servers is the integration between communication and Internet technologies. This has major implications for enabling the creation of many new innovative services for NGN networks. Network Operators/Service providers should also consider the implication of this approch with respect to the balancing of Intelligence at the edge or in the core of the network: Service providers should find their best synergy between edge and core offerings and accepting edge solutions as an opportunity rather than a threat. The terminals emerging support this edge model and will enable the provision of many new and innovative services.

This evaluation has shown that as well as application servers, media servers are also a core component of NGN architecture XML technologies, for example VoiceXML, are also contributing to the integration of communication and Internet technologies. Concerning service creation, development of NGN services is made accessible to a broad public of application developers and in many ways is very close to the web and IT developer community approach, thus helping to enhance the productivity of application development, reducing the time to market of new services and on average only a couple of weeks, and even days, in some cases are required to develop new applications.

Use of open API, Java and XML based scripting languages are paving the way to broaden up the developers community of new and advanced telecommunication services. This will ease the service creation process diminishing time-to-market for the new services.

Acknowledgement

The information presented in this paper is partially based on the work done in the EURESCOM project P1109 "Next Generation Networks: the service offering standpoint" [1]. The authors would like to thank all EURESCOM P1109 participants.

6 References

[1] Eurescom Project P1109 Next Generation Networks: The services offering standpoint. On-line at http://www.eurescom.de/secure/projects/P1100-series/P1109/P1109.htm

[2] Falcarin, P., Licciardi, C.A., Analysis of NGN service creation technologies. In IEC Annual Review of communications, volume 56, on publication in June 2003.

[3] P. Lago, C.A. Licciardi, G. Canal, A. Andreetto, An architecture for IN-Internet hybrid services. In Computer Networks Journal, Special Issue on Intelligent Networks and Internet Convergence, T. Magedanz, H. Rudin, I. Akyildiz (Eds.), Elsevier, Vol. 35(5), April 2001, pp. 537-549

[4] Licciardi, C.A., Falcarin, P., Next Generation Networks: The services offering standpoint. In Comprehensive Report on IP services, Special Issue of the International Engineering Consortium, October 2002.

[5] Handley, M., Schulzrinne, H., Schooler, E. and Rosenberg, J., SIP: Session Initiation Protocol, RFC 2543, March 1999.

[6] Andreetto, A., Licciardi, C.A., Falcarin, P., Service opportunities for Next Generation Networks, In Proceedings of the Eurescom Summit 2001, Heidelberg, Germany, November 2001.

[7] The 3rd Generation Partnership Project (3GPP) Open Services Architecture (OSA). On-line at http://www.3gpp.org.

[8] SCML, Next Generation Service Creation Using XML Scripting Languages, John-Luc Bakker, Ravi Jain, 2002. On-line at www.argreenhouse.com/papers/jlbakker/bakker-icc2002.pdf

[9] SIP Servlets specification. On-line at http://www.ietf.org/internet-drafts/draft-peterbauer-sip-servlet-ext-00.txt

[10] Lennox, J. and H. Schulzrinne, "CPL: A Language for User Control of Internet Telephony Services". On-line at http://www.ietf.org/internet-drafts/draft-ietf-iptel-cpl-04.txt

[11] JAIN Java Call Control (JCC) API, Java Specification Request (JSR) 21", 2001. On-line at http://jcp.org/jsr/detail/21.jsp.

[12] Voice Browser Call Control: CCXML Version 1.0. W3C Working Draft 21st February 2002: http://www.w3.org/TR/2002/WD-ccxml-20020221/

[13] The JAIN APIs: Integrated Network APIs for the Java Platform. (White Paper), Jun. 2000. On-line at http://java.sun.com/products/jain

[14] Java™ 2 Platform, Enterprise Edition Specification Version 1.3. On-line at http://java.sun.com/j2ee/docs.html

[15] VoiceXML specification. On-line at http://www.voicexml.org

[16] UDDI Technical White Paper, http://www.uddi.org/

[17] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[18] Simple Object Access Protocol (SOAP) 1.2 specification. On-line at http://www.w3.org/TR/SOAP

[19] Common Object Request Broker Architecture (CORBA) specification. On-line at www.corba.org

[20] XTML (extended Telephony Mark-up Language). On-line at www.pactolus.org