Open Access Standards for Telecom Service Capabilities

Yang Yong, Jia Xia, Dong Zhenjiang
(ZTE Corporation, Shenzhen 518057, P. R. China)

Abstract:
Among the open access standards for telecom service capabilities, Java Community Process (JCP) and Parlay series are two mainstream standards, which provide service capability opening standards at different levels for different user objects. The JCP specifications include Java Specification Request (JSR) 21, JSR 32, JSR 116 and JSR 289 especially for Java application developers, while Parlay brings out specifications including Parlay and ParlayX. The service capability open technologies feature different benefits and vitality due to their diversified implementations. As the community of service developers is continuously growing and the demands for integrated service development become more and more manifest, fast, effective and easy-to-use open access technologies for telecom service capabilities have become a very important research subject.

Open access to telecom service capabilities has long been a hot topic in the development of telecom technologies[1-7]. Its objective is to open the telecom capabilities to allow easier and more effective development of telecom services, thus enriching the value added telecom services and promoting the development of telecom business.

This article analyzes the different modes and specifications for developing services and open access to service capabilities.

1 Computer Telephony Integration (CTI)

CTI technology[8] for open access to telecom service capabilities can be traced back to the traditional Intelligent Network (IN) and Private Branch Exchange (PBX) based CTI technology. CTI has two major specifications. One is the Telephony Application Programming Interface (TAPI), a product of Microsoft and Intel. It provides a set of Application Programming Interfaces (API) for programming and supports CTI applications on the Windows platform. Its advantages include the connection between Windows-based applications and telephony system. The other is the Telephony Service Application Programming Interface (TSAPI) created commonly by Novell and AT&T. Thanks to the participation of AT&T, TSAPI is perfectly compatible with the existing telephony exchanges.

As a functional supplement for PBX, CTI based call center service has been widely developed and used.

2 IN

The purpose of IN[9] is not only providing a variety of services but also releasing new services in an easy, fast and cost-effective way. As a result, IN provides users with new services by using a brand-new method, that is, establishing a central Service Control Point (SCP) and database, and then setting up a centralized service management service and Service Creation Environment (SCE).

IN defines a complete overall service architecture, which consists of Service Switching Point (SSP), SCP, Service Management Point (SMP), Service Data Point (SDP) and SCE. It is an architecture to generate and provide telecom services and a system to create and realize various new services in a fast, easy, flexible and effective manner.

The system aims to serve all networks. In other words, it provides services for not only the existing Public Switched Telephone Network (PSTN), Packet Switched Public Data Network

(PSPDN) and Narrowband Integrated Services Digital Network (N–ISDN), but also the Broadband Integrated Services Digital Network (B–ISDN), Public Land Mobile Network (PLMN) and Internet.

IN aims to provide various new supplementary services in a fast, easy and flexible manner through service–independent standard communications between functional blocks/entities and by taking advantage of existing resources. The goal will be reached progressively.

The international and domestic INs under construction are at Capability Level 1 (CS–1) now, where the IN service provisioning is limited to PSTN and N–ISDN and within the same network. However, users have demands for service management and service creation, along with the network development and because of the complexity of the actual network operation. So, many new services are added to CS–2, including the inter–network interconnection service, call party handling service (such as Call Forwarding and Call Waiting), terminal mobility and more.

CS–1 and CS–2 Recommendations have been standardized, and the ITU–T is setting about the studies of CS–3. In addition to CS–3, the IN Long–Term Architecture (LTA) is also under study. LTA is put forward to cater for the rapid progress of technologies, fast growth of users' service demands, interconnection between various services and competition mechanism introduced into the communication market. Its goal is to introduce a new communication network control and regulation system that flexibly adapts to new technology development and meets potential service demands.

So, the development of IN is based on the development of services, and expands gradually to the domains of mobile communication and broadband communication. It shall also be integrated with the telecom management network to provide a more flexible communication system suitable for newer technologies.

The recent development is the use of integrated IN system, in which the same IN network can be connected with different protocols including GSM, CDMA, PSTN, Parlay and HTTP, and the service logic processing is independent of network type.

The birth of IN enables separate call control from services and greatly facilitates the development of services. In traditional IN system, however, the SCE is still tightly bound with the SCP, and there is neither uniform standard nor openness. Generally, only the developers of telecom vendors develop intelligent services using their own SCE.

3 Java Community Process (JCP)

JCP is a Java–based development organization engaged in the establishment of Java specifications. The early Java–based CTI specifications is the Java Telephony Application Programming Interface (JTAPI) developed in 1996 by a workgroup of researchers from Intel, Lucent, Nortel Networks, Novell and Sun Microsystems.

JTAPI is a Java–based application programming interface for computer telephony application. It consists of a set of language packs, with the core pack providing a basic framework for simple telephony process, for example, making call, answering call or hanging up, and other extension packs providing additional telephony features. JTAPI can be used in different computer platforms, just like the TAPI. The relationship between them is something like the relationship between Open Database Connection (ODBC) and Java Database Connection (JDBC).

The JCP organization continues to perfect the Java–based specifications and defines them as the JCP standard, releasing in succession many Java specifications for opening telecom service capabilities[10–11]. The following sections introduce the Java specifications in detail.

3.1 Java Specification Request (JSR) 21

The JSR 21 specification[12] is called "Java APIs for Integrated Networks (JAIN) Java Call Control Application Programming Interface" in full, which is a Java interface used to establish, monitor, control, manipulate and release communication session under the PSTN/packet/wireless environment. It provides third–party applications with network element capabilities including core network and peripheral devices. Java Call Control (JCC) can be triggered or invoked during a session setup, basically similar to the mode of service invocation in IN or Advanced Intelligent Network (AIN). Therefore, JCC allows programmers to develop applications that can run on any platform supporting those APIs. And, service providers can fast and effectively provide service subscribers with the services of their own or by buying services from a third party. APIs defined in the JCC specifications are inherited from JTAPI.

JCC API does not open the telecom network signaling architecture but encapsulates network capabilities, which allows the capabilities to be represented and used in a safe, manageable and chargeable manner using a visual object technology. This method allows independent service developers to develop telecom service with no negative impact on the network security and reliability. JCC APIs are defined with related and interactive object sets, which modularize different physical and logical elements or related functions involved in a session. The applications interact with the objects through object–oriented procedure charts. JCC APIs can control voice calls, as well as data and multimedia sessions.

Structurally the JCC APIs fall into three types:

(1) Basic call control: This Java pack includes the basic tools for initiating and answering calls.

(2) Core call control: This Java pack includes the tools for monitoring, initiating, answering, processing and manipulating calls, as well as some tools that invoke other applications and return results during the process of a call. It can meet the requirements of most basic call and value–added service implementations.

(3) Extended call control: This Java pack provides some extended granular call control functions. Unlike Java Coordination and Transaction (JCAT), in particular, JCC supports all universal AIN applications and the integration with other voice/data and next generation services.

The applications developed with the above packs can be executed on the

switch platform or across multiple platforms in a collaborative and distributed way.

JCC API provides only the Java API definitions, while the interface implementations must be made by the equipment vendor. This set of APIs shields the network transfer layer and fits for any type of network including PSTN, IP or wireless network. Also, it does not concern the underlying communication protocols or signaling, which can be Media Gateway Control Protocol (MGCP)[13], Session Initiation Protocol (SIP)[14] or Signaling System No.7 (SS7). The application developers do not need to have the network information.

Therefore, the key purpose of JSR 21 is to enable third–party Java developers to develop call services. However, the API defined in the specification is a bottom layer–oriented operation interface for the call control signaling, and the application developers must have basic telecommunication knowledge. In addition, the SIP–based applications developed with JCC API are usually limited by the expandability of API because of the high expandability of SIP.

3.2 JSR 32

The full name of JSR 32[15] is JAIN SIP API Specification, which is the interface specification defined and developed by the JCP organization for SIP applications, providing the developers with RFC 3261–compatible Java–based standard SIP service interfaces.

The JSR 32 specification standardizes the interface to the generic transactional model defined by the SIP protocol, providing access to dialog functionality from the transaction interface. The architecture is developed for the J2SE environment therefore is event–based utilizing the Listener/Provider event model. It defines various factory classes for creating Request and Response messages and SIP headers. It defines universal interface for each supported header, which can be added to Request or Response messages respectively. These messages are passed via a transaction to the Listener/Provider model to listen to the coming events, including the response to a request or a new request. In addition, the JAIN SIP is designed to be extendable with universal extension header interfaces defined, so that the unsupported SIP header domain can be used in the processing specification.

It is notable that the default handling for SIP message resend is dependent of the application type, and all resend operations are processed by the protocol stacks in the User Agent (UA).

Since JAIN SIP API is a complete definition of the SIP standard, any SIP–based program may utilize the JAIN SIP API as a standard Java interface into any JAIN SIP certified stacks. This means the versatility in SIP stack implementation can be achieved by using the JAIN SIP API for application servers, SIP telephones, gateways and gateway controllers, SIP servers, SIP–based services, SIP billing solutions, development kits, SIP test tools, SIP user agents and SIP network administrators. In a typical SIP network, the JAIN SIP interface is used as an agent server, media gateway and client. JAIN SIP APIs cover more than those required by user agent or client software.

JAIN SIP APIs provide basic classes in four packs, including:

(1) javax.sip: This package contains the key interfaces of the basic architecture provided from the view point of the application developers and protocol vendor.

(2) javax.sip.address: This package contains the interfaces used to present the address components in the SIP.

(3) javax.sip.header: This package contains all the headers interfaces supported by this specification.

(4) javax.sip.message: This package contains the interfaces used to present the SIP message body.

Compared with the JSR 21 specification, JSR 32 specification aims specially at the SIP networks such as next generation network or IP Multimedia Subsystem (IMS). They also provide the developers with a Java SIP stack and related interfaces. These interfaces are protocol layer interfaces, while in the JSR 21 specification they are interfaces at the functional operation level, much higher than the level with the JSR 32 specification. The interfaces provided by JSR 32 allow developers to get or to manipulate all fields including SIP header domain and all SIP message contents including SIP message body.

3.3 JSR 116

The full name of JSR 116[16] is SIP Servlet Specification V1.0, which is a set of SIP container based SIP application development standards.

SIP Servlet is a Java–based application component, which is managed by SIP container and which also implements SIP signaling processing. Just like other Java–based components, Servlet is a platform–independent Java class and can be dynamically loaded to run on a Java–based SIP application server. Containers are the extension of the server that provides the Servlet function. With the Servlet containers, Servlet interacts with clients through the exchange of request messages and response messages. SIP Servlet container is a part of the application server, receiving and transmitting request/response messages for the network layer services. It determines which application is triggered by the received SIP message and in what order to trigger. Meanwhile, SIP Servlet container has the Servlet lifecycle management function, and is also responsible for the support of User Datagram Protocol (UDP), Transfer Control Protocol (TCP) as required by the SIP specification for all SIP network elements, optional support of Transport Layer Security (TLS) and Stream Control Transmission Protocol (SCTP), etc. SIP Servlet is primarily used to develop SIP–based applications. Currently, the SIP–based applications are the call type applications, instant messaging, online SIP short message, as well as SIP short messaging deriving from SIP Message method. The interface specification used to develop SIP applications provides developers with RFC 3261–compatible Java–based standard SIP service interfaces.

Within JSR 116, the major functions of SIP Servlet container are application management, SIP message processing and tool functions (SIP session and application session, SIP factory and agent). JSR 116 is an SIP application development tool that is specific to SIP and provides container type SIP application management. In terms of interface encapsulation degree, it is a

kind of interface specification between JSR 21 and JSR 32. It provides not only method–level operation interface but also protocol–layer data operation interface. In terms of application development, its flexibility and usability are perfect. JSR 116 is the standard version 1.0 of the SIP container product, and JSR 289 Version 1.1 has been published. Compared with JSR 116, there is no change with the framework but improvement on some interface definitions. The latest SIP Servlet specification is Version 1.1. Refer to JSR 289[17] for more details.

In addition, the SIP Servlet specification defines only interfaces that are based on the SIP Servlet to develop SIP applications. As we know, most telecom applications involve voice, number receiving, conference and other basic telecom function, while the SIP Servlet specification including JSR 116 and JSR 289 has not defined interfaces related with those functions. Therefore, the JCP organization published in 2007 JSR 309[18]—Java Media Server Control, which defines the functional interfaces related with media server control to work with the Java–based SIP application development.

In addition to the above specifications, JCP also defined a Java specification for service execution environment, JAIN Service Logic Execution Environment (SLEE) V1.0. JAIN SLEE is an integral part of the JAIN API set. It is located in the core as the logic execution environment for applications. Refer to JSR22[19] for details.

4 Parlay Related Specifications

The Parlay organization was established in 1999 as a non–for–profit organization with participations from 65 telecom and IT companies, dedicating to define Parlay APIs as a set of open, technology–independent and expandable APIs, so that third–party service developers and independent software vendors can develop services with the Parlay APIs. Till now several versions of Parlay specifications have been published. Open Service Architecture (OSA) is referenced in the 3GPP and 3GPP2 mobile service architecture, while Parlay is just the API part in OSA.

The definitions of the Parlay APIs[20] are described with the Interface Description Language (IDL), and the APIs are implemented with the distributed Corba midware technology.

The Parlay organization makes researches on the part of open interface and cares not about the basic telecom network structure and technology. The interface is located between the service provisioning network part and the core network part. While the Parlay organization works on the research of Parlay 2 standards, 3GPP and ETSI launch the researches on 3G–network–based application development APIs. The research work is overlapped to a large extent, and 3GPP and ETSI find soon that Parlay can be used in their 3G network APIs. So, Parlay is introduced into the 3GPP/ETSI standard framework and named as OSA, and 3GPP/ETSI provides further complements for the Parlay standards. Now 3GPP has published API standard Release 5, and ETSI and Parlay Group have also published related versions, with the latest version, Parlay 6.0.

When the Parlay API is upgraded to Parlay 4.0, the Parlay organization found that the Parlay specifications are quite complicated for IT R&D personnel, and it was not easy to work with the Parlay protocol specifications to develop applications on telecom networks. For this reason, 3GPP puts forward the ParlayX specification[21]. The ParlayX protocol further abstracts the APIs on the basis of the original Parlay API protocols, and uses Web–service–based Web Service Description Language (WSDL) to describe the APIs, so that the IT R&D personnel is provided with a clearer, cleaner, abstract and easy–to–understand telecom service development interface. In this way, the IT R&D personnel do not need professional knowledge of telecom network and can develop and apply the next generation network services with the ParlayX protocol interface. It is a promotion to the development of the next generation network services.

After Parlay 4.1 has been developed to ParlayX, it gradually divides into two branches. The original Parlay specification continues its progress till the latest 6.0, while the ParlayX specification also evolves, from version 2.0, 2.1, 2.2 till the latest ParlayX 3.0.

4.1 Parlay Specification

The Parlay specification defines multiple sets of Service Capability Features (SCFs), each of which has a set of APIs for opening service capabilities. As shown in Figure 1, the logical structure of Parlay/OSA consists of four major parts:

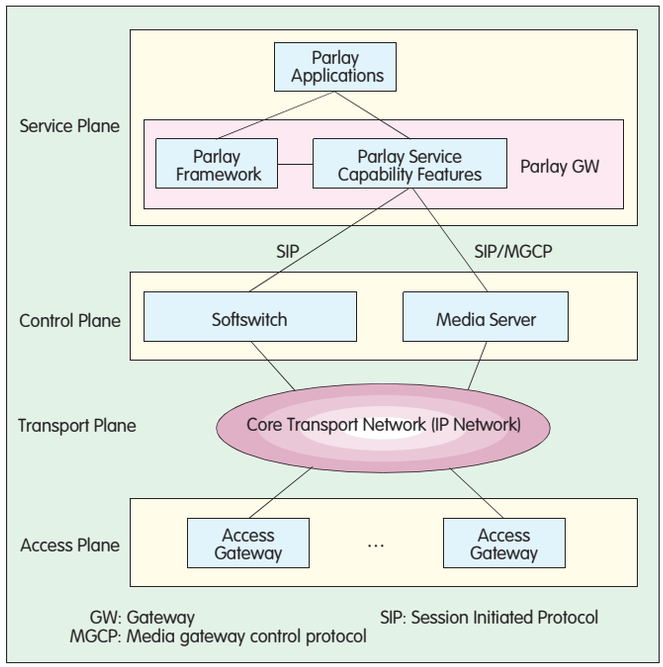
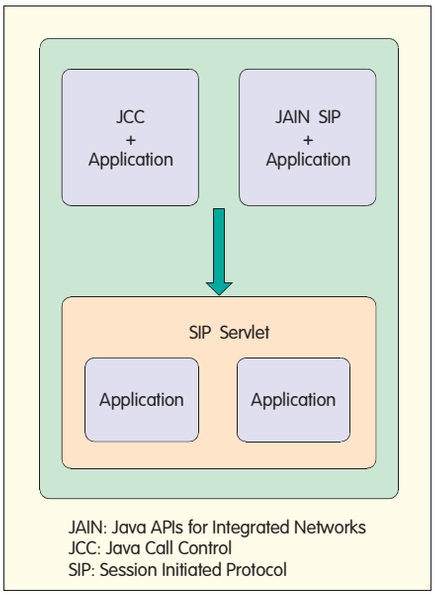


Figure 1. ▶ Logical structure of Parlay/OSA.



▲Figure 2. Comparison between JCC&JAIN SIP application mode and SIP Servlet application mode.

Parlay application, Parlay/OSA framework, Service Capability Server (SCS) and core network. A Parlay Gateway is made up of the framework and service capability servers.

Parlay 4.0, taken as an example, defines 11 SCFs in total. They are call control, user interaction, mobility management, terminal capabilities, data session control, normal message, connectivity management, account management, billing, policy management, representation and availability management.

The open structure of 3G network is a prerequisite for Parlay Gateway to control network resources. The Call Session Control Function (CSCF) equipment is the core in the 3G IMS network. It is dependent of the underlying bearer protocol, implements call control, media gateway access control, resource allocation, protocol processing, routing and other functions, and can provide users with the services available in the existing networks.

The applications that are developed on the basis of the Parlay Gateway provide a residential and execution environment for service logics, and the development platform is provided for the third-party service developers through the APIs available with the Parlay Gateway. Parlay Gateway is the principal in the service provisioning plane and also the core for service provisioning and development. It helps the service plane provide rich services with various resources of the underlying network. This architecture enables separate services from call control, and separate call control from bearer, so as to implement relatively-independent service function and upper-layer services unrelated to underlying network, making services available in a flexible and effective way.

4.2 ParlayX Specification

The Parlay organization simplifies the Parlay APIs to provide those APIs for third parties in forms of Web services, resulting in the ParlayX specification. The first version of the ParlayX 1.0 was officially released in April 2003, and the latest specification version is ParlayX 3.0.

Take the ParlayX 2.0 specification as an example, where the multiple sets of service capabilities are defined, including third-party call, call notification, short message, multimedia short message, voice call, terminal status, terminal location, account management, call processing, payment, multimedia conference, address list management and presence.

The ParlayX makes high-degree abstraction and encapsulation for the original Parlay APIs and defines a set of powerful yet simple, abstract and imaginative telecom capability APIs, so that the telecom developers and IT developers can understand and grasp them quickly and then develop creative telecom application software.

The ParlayX works in forms of Web services, and the openness of the Web services makes the ParlayX more acceptable and recognizable by IT developers. The interaction between the applications developed with ParlayX APIs and the server implementing ParlayX Web service (also called the ParlayX gateway) is implemented through the Extensible Markup Language (XML) based message exchange. The message exchange is initiated by the application and follows the synchronous Request/Response model. The response from the ParlayX Web service serve to the application is optional, depending on actual requirements.

However, asynchronous messages must be defined for the message notification services, where application server acts as the passive party, to implement the message transfer from ParlayX gateway to applications.

The encapsulation of the ParlayX APIs is much higher than that of the Parlay APIs. For example, with the Parlay APIs, at least three times API invoking is needed at the application side to initiate a call: createCall→ routeReq (A) → routeReq (B). With the ParlayX, only one time API invoking "makeACall" is enough. Such high encapsulation simplifies the efforts greatly for the developers.

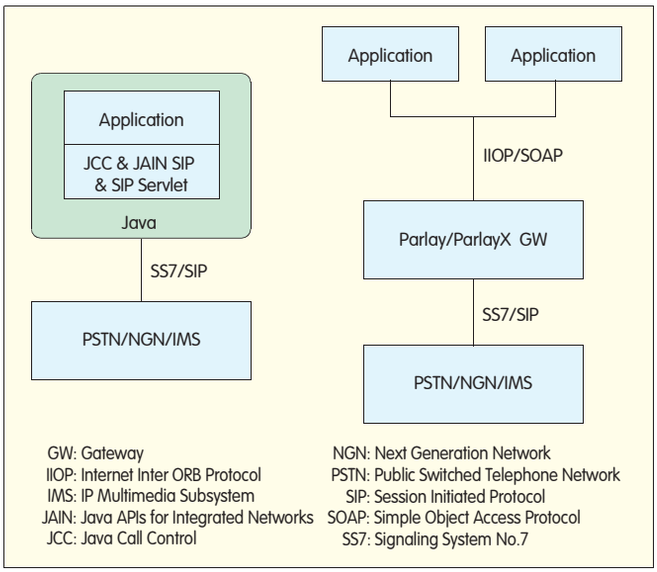
However, insufficient continuous call control, low user interaction capability and lack of user authentication are challenges for the ParlayX. Relevant organizations are working on its improvement.

5 Comparison of Service Capability Open Access Technologies

First, JSR 21 and JSR 32 are used for the independent development of applications, while JSR 116, due to the introduction of container, allows bearing the operation of multiple applications, as shown in Figure 2. Meanwhile, the container in JSR 116 can be combined with the HTTP container to develop some Web-based integrated SIP applications. Another difference between JSR 32 and JSR 116 lies in JAIN SIP defined by JSR 32 is J2SE application oriented while the SIP Servlet specification defined by JSR 116 is J2EE application oriented.

Second, the operation mode for the JCP series based applications is different from Parlay based applications. In terms of application deployment and running mode, the operation of Parlay/ParlayX applications is in a distributed mode since the Parlay/ParlayX is implemented on the basis of distributed technologies.

For the applications that are developed on the basis of the JCC, JAIN SIP and SIP Servlet specifications, however, the API implementations and applications are executed by the same Java virtual machine, which are bound together physically. The difference between them is illustrated in Figure 3.



◀Figure 3. Operation mode of JCP applications and Parlay applications.

As shown in Figure 3, the applications based on JSR series specifications interconnect with core network element in their deployment, and the interface in between is standard SS7 or SIP. In contrast, the applications based on Parlay/ParlayX specifications interconnect with the IIOP/SOAP or Parlay/ParlayX gateway in their deployment, adapted into standard SS7 or SIP through the Parlay/ParlayX gateway.

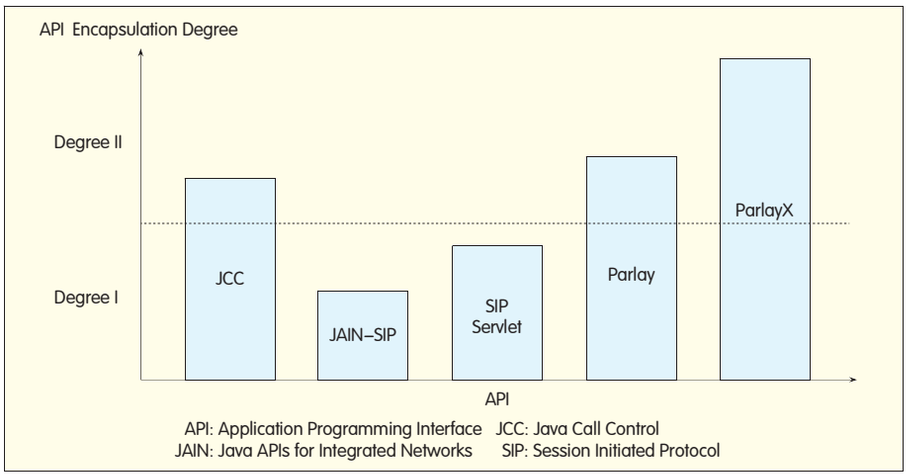
In contrast, the distributed deployment is more advantageous than the centralized application mode. But, the shortcoming of distributed deployment is that the developers have to grasp the Corba or Web service technology in addition to familiarity with basic call signaling knowledge, raising higher requirements for the developers to some extent. Meanwhile, the JCP series specifications (JCC, JAIN SIP and SIP Servlet) define only the interface for application function development but do not tap into the authentication access, access control and other functions for the applications. So, it is applicable for only the application development in some trustable domains. The Parlay specification provides not only the basic call capability but also a complete development, operation and management system for the applications, including authentication access, access control, and lifecycle management. In particular, the ParlayX also has some basic functions on service access and control through the extension of SOAP header. Most things have two handles; simplicity and flexibility are usually mutually exclusive.

Third, in terms of technological implementation, the JSR series define Java API implementations which inherit the across-platform characteristics with Java language. The Parlay API implementation is based on Corba midware technology, which is a platform across technology, independent of the operating system and programming language. In other words, even when the application and service are running on different operating systems, it is possible to use a different programming language to implement API functions and invoking. ParlayX is implemented on the basis of Web service technology, which features more advantageous openness and can be implemented with different programming languages. For application developers, an interface technology across platforms and independent of language is definitely more acceptable.

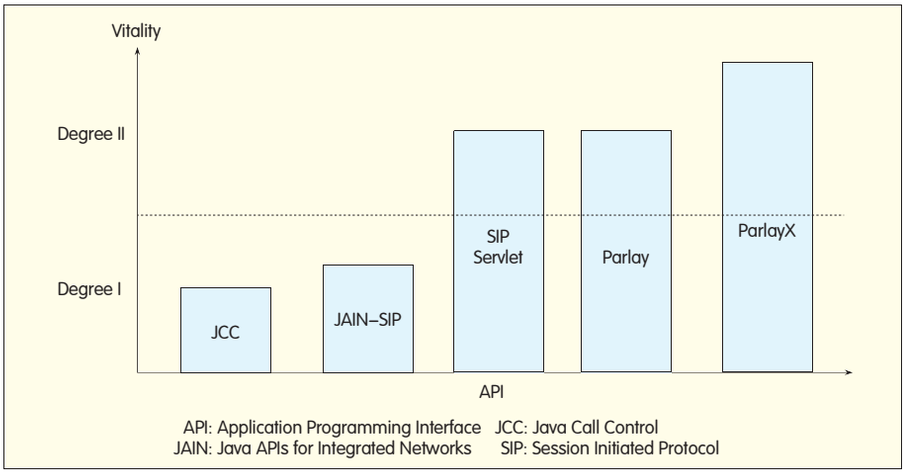
Undoubtedly different service capability opening technologies shall not be simply judged, and each of them has its own merits. Service developers at different levels can choose a suitable technology to develop telecom applications for specific service requirements.

The specifications are also compared in terms of API encapsulation degree, as shown in Figure 4. At a higher level, interface encapsulation falls into two categories: Operation level API encapsulation and protocol layer API encapsulation. JCC and Parlay/ParlayX provide operation level APIs, with higher degree API encapsulation and API implementation independent of specific protocol. JAIN SIP and SIP Servlet provide protocol layer APIs and API encapsulation specific to SIP, which, to some extent, is equivalent to a Java version SIP protocol stack. So, the SIP Servlet provides higher level APIs than JAIN SIP specification. For the application developers, it is easier to use interface technology of higher encapsulation degree in their application development.

Finally, the specifications are compared in terms of vitality, as shown in Figure 5. The specifications established by two organizations JCP and Parlay are continuing their growth. For the time



▲Figure 4. Comparison of API encapsulation between different specifications.



▲ Figure 5. Vitality comparison between the specifications.

being, there are not many supports or positive response to JCC and JAIN SIP from the equipment vendors, while SIP Servlet product has won clear supports from quite some vendors. And, many vendors are supporting Parlay/ParlayX, in which the Parlay Gateway products from Ericsson are being used most widely. Vendors' supports for ParlayX are even more popular.

On the whole, vendors' supports for Parlay/ParlayX are better than those for the JCP specification. However, the JCP organization has researched service capability openness and also made in–dept study on service execution environment, resulting in the specifications like JAIN SLEE. The specifications stimulate the development and popularization of JCC, JAIN SIP and other specifications, which will be specially introduced in future.

6 Conclusions

In terms of the implementation of open access technologies for telecom service capabilities, especially the interfacing techniques, different interfaces are provided with different technologies. As mentioned above, higher encapsulation rate allows easier interface–based applications; and lower encapsulation rate allows interface functions with higher flexibility and interface–based application implementation with higher complexity. Therefore, technologies cannot be simply classified as good or not. Actually, the application developers on different levels with various demands should be provided with matching open service interfaces to fulfill open access to service capabilities at different levels and granularity.

The cycles of service development are shortened continuously along with the gradual opening of service capabilities. The groups of service development personnel are expanded continuously, and the coupling relationship and interactions between services are getting more and more complex. Therefore, how to achieve the orderly monitoring and management between services, between service capabilities and between service and capability are important issues for research on the next generation of service creation environment and execution environment based on the distributed, open Information and Communication Technology (ICT) convergence environment.

References

[1] 张云勇, 刘韵洁, 张智江, 等. 下一代网络业务开放的相关问题 [J]. 电信科学, 2004, 20(1): 41–46 .
[2] 强磊, 陈卉. NGN的统一业务开放平台与增值业务的创新 [J]. 中国数据通信, 2004, 6(5): 71–74.
[3] 刘韵洁. 下一代网络的发展趋势—融合与开放 [J]. 电信科学, 2005, 21(2): 5–10.
[4] FALCARIN P, LICCIARDI C A. Analysis of NGN service creation technologies [J]. IEC Annual Review of Communications, 2003, 56(6): 100–110.
[5] LICCIARDI C A, FALCARIN P. Next generation networks: the services offering standpoint [R]. Eurescom Project P1109. 2002.
[6] LAGO P, LICCIARDI C A, CANAL G, et al. An architecture for IN–Internet hybrid services [J]. Computer Networks Journal, 2001, 35(5): 537–549.
[7] BOETSELAARS L, et al. CD–ROM: enabling technologies for IN–Internet integration [R]. Eurescom Project P909 Deliverable 4. 2001.
[8] 叶飞, 傅海阳. CTI技术及其在现代通信中的应用 [J]. 电子工程师, 2001, 27(6): 36–38, 41.
[9] 张雪丽. 智能网标准概况 [J]. 电信工程技术与标准化, 2004(1): 34–38.
[10] JAIN. A set of Java APIs for integrated networks [R]. Telcordia Technologies Inc. 1999.
[11] The JAIN APIs: Integrated network APIs for the Java platform [R/OL]. White Paper. 2000. On–line at http://java.sun.com/products/jain.
[12] JSR21– JAIN Java Call Control (JCC) Application Programming Interface (API), Version 1.0 [S]. JCP. 2001.
[13] ARANGO M, DUGAN A, HUITEMA C, et al. Media Gateway Control Protocol (MGCP), Version 1.0 [R]. RFC 2705.1999.
[14] ROSENBERG J, SCHULZRINNE H, CAMARILLO G, et al. SIP: Session Initiation Protocol [S]. RFC 3261. 2002.
[15] JSR32–JSIP API Specification v1.2, Final Release [S]. JCP. 2006.
[16] JSR116– SIP Servlet API, Version 1.0 [S]. JCP. 2003.
[17] JSR289–SIP Servlet Specification v1.1, Final Release [S]. JCP. 2008.
[18] JSR309–Java Media Server Control [S]. JCP. 2008.
[19] JSR22–JAIN SLEE 1.0 Specification, Final Release [S]. JCP. 2004.
[20] ETSI ES 201 915–1 v1.1.1. Open Service Access; Application Programming Interface. Part 1–Part 12 [S]. 2001.
[21] Parlay APIs 4.0: ParlayX webservices [R]. White Paper. Parlay Group, 2002.

Biographies

Yang Yong

Yang Yong received his doctoral degree from Southeast University, and now is a senior engineer at ZTE Corporation. He has been engaged in the study of Intelligent Networks (INs) and Parlay/ParlayX gateways. His current research interests include service delivery platforms, evolution of NGN and IMS, and multimedia techniques. He has applied for seven patents and more than ten papers.

Jia Xia

Jia Xia received her master's degree from Dalian University of Technology. She is responsible for the pre–research on next generation service networks at ZTE Corporation. She has been engaged in the research and development of fixed IN services, broadband services and IMS services. Her research interests include service networks, SDP, SOA, service engine and mobile Internet. She has published five papers on service network evolution.

Dong Zhenjiang

Dong Zhenjiang received his master's degree from the Harbin Institute of Technology, and now works for ZTE Corporation. He has been engaged in the development, design and planning of switch and IN networks. His research interests include SDP, P2P, service engine, 3G services and ICT. He has participated in or presided over multiple research projects sponsored by the National Development and Reform Commission of China. He has published more than 10 papers on value–added services.