

# The Development of a Structured Approach to Service Provisioning in a Parlay Environment

B. Fricke and H.E. Hanrahan  
Centre for Telecommunications Access and Services (CeTAS)<sup>1</sup>  
University of the Witwatersrand  
Johannesburg, South Africa  
e-mail: {b.fricke, h.hanrahan}@ee.wits.ac.za

**Abstract**—The OSA/Parlay architecture supports the development of applications that utilise network resources through standardised, open APIs. However, Parlay provides no guidance as to how the services, or the service domain, should be structured. Service developers and enterprise operators have to implement proprietary approaches to service provisioning, and little work has been done in the industry towards a standardised approach to service domain structure. Additionally, the operational environment in which the service domain operates is not conducive service provisioning, limiting the scope and capabilities of services that can be provisioned.

This paper presents a structured approach to service provisioning in a Parlay environment. This entails the development of a structured service domain architecture, which advocates functional layering and software reuse. Changes to the underlying network are also advocated, so that the service domain can be served more effectively: a new approach to service signalling and service invocation is presented, which defines a direct communication path between terminals and the service domain. Finally, the relocation of the connectivity control functionality from the underlying network to the service domain is proposed.

**Index Terms**— connectivity control, OSA/Parlay, service domain, services, signalling, structure

## I. INTRODUCTION

THE goal of Parlay is the specification and realisation of an open, technology-independent Application Programming Interface (API) for telecoms networks [1]. The Parlay API enables network operators, independent software manufacturers and service providers to develop products and services that use the functionality of existing networks.

The Parlay APIs are shown in the context of the service domain, and the network as a whole, in figure 1. There are three horizontal layers: the service domain, the Parlay gateway and the bearer network. Parlay provides an abstraction layer between the telecoms services (in the service domain) and the bearer network. Within the Parlay gateway there are a number of Service Capability Features (SCFs). SCFs provide abstractions of the functionality

offered by the network, accessible via the OSA/Parlay API [2].

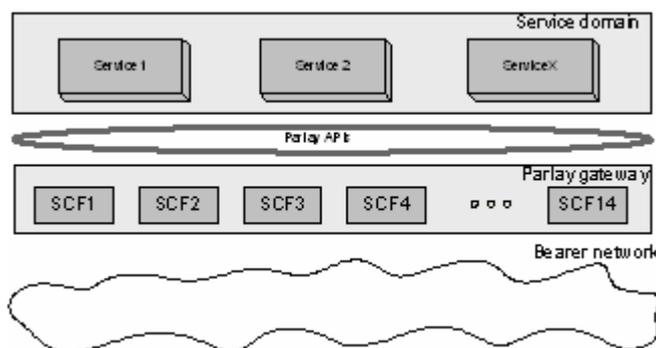


Figure 1: Parlay and the service domain

Operators are looking to the service domain as a key element in their overall business strategy because it has such a high impact on their revenues [3]. Since value-added services are responsible for revenue generation, and which ultimately determine the contribution of a network, networks should appropriately have a service-centric orientation, providing an environment conducive to the provisioning of advanced services.

However, many legacy and current telecoms networks have a more transport-layer orientated approach, which limits both the scope and capabilities of services that networks can provide users. This approach makes service development and deployment time-consuming and complex, and is unable to meet the demands of any modern service provisioning environment.

To introduce a service-centric approach into present transport-layer oriented networks requires network alterations in two areas. The first, and most obvious, concerns the service domain itself. A service-centric network requires a service domain which enables advanced services to be developed and deployed in a simple manner by a wide range of application developers. Secondly, the underlying bearer network should be designed to provide an environment conducive to service execution.

This paper presents a structured approach to service provisioning in a Parlay environment. Proposed developments primarily concern the service domain; additionally, changes concerning the network outside of the service domain are proposed, toward the end of a more conducive environment for service provisioning.

<sup>1</sup>The Centre is supported by Telkom SA Limited, Siemens Telecommunications and the THRIP Programme of the Department of Trade and Industry

## I. SHORTCOMINGS OF THE PRESENT APPROACH TO SERVICE PROVISIONING

This analysis of the present approach to service provisioning centres around two aspects: the approach to the service domain (using Parlay), and the approach to the telecoms environment in which the service domain operates.

Services and applications in convergent networks contain an increasingly large software part, as opposed to traditional telecoms services where software plays a more modest role [4]. The development of complex software systems is expensive and error-prone [5]. The OSA/Parlay specifications provide little detail as to how the service domain should be structured, or how Parlay services should be implemented; service developers and enterprise operators need to adopt proprietary solutions. The lack of standardisation increases the skill-set required to develop services, and increases service development times.

Parlay reduces integration costs by having the overall network integration done once, between the Parlay gateway and the operator's network. Since all applications communicate via the Parlay gateway, they do not need to be individually integrated to the operator's network [6]. Although Parlay allows service providers to reuse their services on multiple networks, it does not provide any guidance regarding higher-level reuse, such as reuse at the application level, or reuse of component compositions [4].

The Parlay standards imply that the control of the basic telecoms service of providing end-to-end user connectivity is still located in the bearer network, whereas all other services are located in the service domain. A service-centric approach would allow for the control of bearer connectivity in a centralised location, alongside other services provided to customers, rather than in the underlying network.

The invocation of services in the service domain can be achieved through either of two scenarios. Firstly, a service could be triggered by communication in the bearer network meeting a trigger condition, and causing the service to be invoked (1<sup>st</sup> party invocation). Secondly, the original communication could be initiated by the service, which was invoked by a 3<sup>rd</sup> party. For 1<sup>st</sup> party service invocation, the terminal is oblivious to the service execution, and the present approach to service signalling is adequate. However, 3<sup>rd</sup> party invoked services that require complex interaction with the terminal for their execution require a signalling protocol that is appropriately oriented and powerful. In the present approach, signalling between terminals and the service domain is required to be routed through the bearer network, using simple bearer network protocols, and can not travel to the service domain directly. 3<sup>rd</sup> party services that require complex interaction with the terminal require a signalling protocol that is more powerful. A direct signalling path between the service domain and terminals is required, allowing the use of suitably oriented protocols.

1<sup>st</sup> and 3<sup>rd</sup> party services were defined as having different sources of invocation: the bearer network, or the terminal, respectively. In addition to the terminal, 3<sup>rd</sup> party services can be invoked by other sources as well, e.g. web services. How services should deal with invocation from various sources, each using different protocols, is not covered in the Parlay specification. The potential for various types of

sources to invoke the telecoms services imposes a burden on each and every telecoms service, since each has to be equipped with the ability to process invocation requests of various forms, unless a standardised approach is developed.

## II. FORMAL PROBLEM STATEMENT

The problem that this paper addresses is that the current service provisioning environment is not conducive to the implementation and execution of telecoms services. Specifically:

- The service domain is poorly developed.
- The telecoms environment in which the service domain operates is inefficient, from the perspective of service provisioning.

In the absence of a standardised service domain architecture, network operators need to either develop proprietary service domain architectures, or the operators' service domains remain devoid of structure altogether. This results in service developers needing employ non-standardised and ad hoc approaches to service development and deployment.

Left unattended, the second part of the problem increases the effort required to develop services, and places a cap on the variety and complexity of services that can be successfully provisioned.

This research project tackles both parts of the problem in attempting to redefine the environment in which services operate. Each of the problem parts has a number of sub-problems:

- The service domain is poorly developed
  - The service domain has no solid architecture
  - The services have no common structure
  - No software reuse methodology has been defined
  - No guideline exists concerning how services should handle invocations from multiple sources
- The telecoms environment in which the service domain operates is sub-optimal
  - No mechanism exists for terminals and services to communicate directly
  - The basic connectivity service has no centralised point of control and management.

## III. A STRUCTURED APPROACH TO SERVICE PROVISIONING

The goal of creating an environment conducive to the development, provisioning, invocation and execution of services is achieved through two efforts: firstly, a well-defined service domain architecture is developed, and, secondly, the operational environment of the service domain is modified.

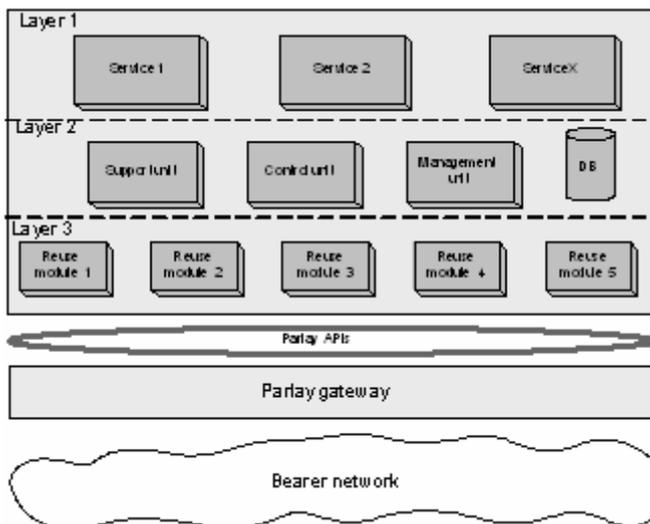
Three principal aspects of the proposed approach to service provisioning are presented next: a structured service domain; a new approach to service signalling; and a new approach to bearer connectivity. While the definition of the structured service domain concerns the service domain explicitly, the new approach to service provisioning also has consequences for the bearer network. The features concerning the bearer network are proposed due to their influence on the contextual environment of the service domain, which exerts significant influence on the service domain itself.

### A. Service domain structure

The first distinguishing feature of the proposed approach to structured service provisioning is based on the recognition that neither services, nor the service domain, have well developed structures in the Parlay standards. The proposed service domain architecture remedies this problem.

Despite services being highly diverse, core capabilities and functionality are often common and can be widely shared among them. Reuse of software building-blocks is essential to leverage the domain knowledge of expert developers, and to avoid re-developing and re-validating common solutions to recurring requirements and software challenges [5]. The proposed service domain architecture employs a software reuse methodology.

In figure 2, the proposed service domain architecture is structured with a layered, hierarchical approach to promote software reuse. The details of the structure of the proposed service domain architecture can be found in section IV.



**Figure 2: A layered service domain architecture employing software reuse**

A properly developed service domain architecture, coupled with a well-defined approach to service structure using a standardised software reuse methodology affords software developers greater ease in developing telecoms services, allows services increased simplicity and efficiency, and increases the capabilities of the services that can be offered.

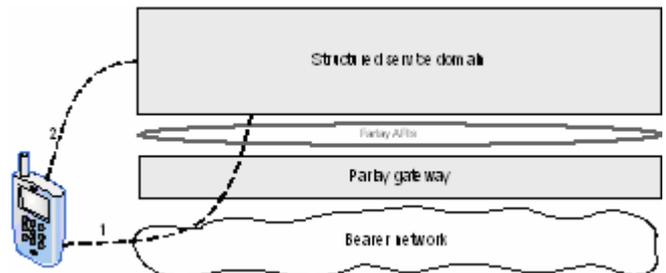
### B. Service signalling

In legacy telecoms networks, all service(s) resided in the bearer network, and the bearer network was solely responsible for the processing of all services. All service signalling was therefore, appropriately, sent by terminals to the heart of the bearer network, where the services were located. The simplicity and nature of the original telecoms services ensured that the use of network-layer signalling was adequate for their invocation and execution.

Due to these origins of the telecoms network, all service signalling in present networks is still sent by terminals to the heart of the bearer network, using bearer network protocols, even though services have now been relocated to the service domain and are far more complex.

This service signalling is no longer efficient for advanced services. The protocols used to achieve signalling in the bearer network are oriented towards establishing bearer connections and setting up media streams, and are not appropriate for complex communication between advanced services and terminals.

Figure 3 depicts the present and proposed approaches to service signalling. In the present approach, all signalling between a terminal and the service domain is forced to travel through the bearer network, and use bearer network protocols. Signals from the terminal are sent to the bearer network, and the bearer network redirects these signals to the service domain. Similarly, if the service wishes to send information to a terminal, the information is sent via the bearer network. This approach is represented by the signalling path numbered 1.



**Figure 3: Approaches to service signalling**

This present approach is cumbersome. Direct communication between terminals and the service domain should be possible, not requiring the oversight of the bearer network. Such signalling is named *application layer signalling*, and is represented by the signalling path numbered 2. Application layer signalling is not limited to the use of simple network protocols. Rather, high-level signalling, oriented to service control, is used, allowing a larger range of more advanced services to be implemented. Application layer signalling is utilised by the proposed approach to service provisioning.

The messages offered on the application layer signalling channel are standardised, and defined by the Application layer API set, discussed in section V.D.

### C. Bearer connectivity control location

The principal objective of telecoms networks, traditionally, was to allow users of the network to communicate in real time. Before the advent of supplementary services, this was indeed the sole goal of telecoms networks. Since bearer connectivity was the only major focus of legacy telecoms networks, the logic and intelligence required to achieve control of bearer connectivity was deeply entrenched and integrated into the heart of the bearer network itself. In contrast, the control of supplementary services is maintained in the service domain.

Due to the origins of telecoms networks, the existing approach to the handling of bearer connections still has all processing, control and management of the bearer connection maintained in the network domain itself. The proposed approach to bearer connectivity advocates that the primary view of the call and the highest level of control of the call are in the application layer (specifically, the service domain).

Having the primary control and view of the bearer connectivity service located in the service domain, and not distributed across numerous network elements, allows a more holistic, centralised approach to connectivity management. Now, when connections are established using 3<sup>rd</sup> party invocation, connectivity control and the connection view are maintained by the service domain. The proposed service provisioning environment advocates this approach to bearer connectivity.

#### D. Establishing bearer connectivity

With control of the bearer connectivity service relocated to the service domain, there are two alternatives concerning how an end-to-end bearer connection can be established. The first alternative is for the originating terminal to establish the bearer connection using the more traditional ‘1<sup>st</sup> party approach’, shown below.

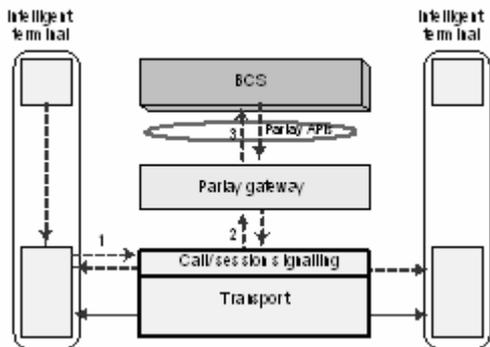


Figure 4: ‘1st party’ approach to connectivity creation

Under this approach, the terminal signals to the network (1) that a connection is required, which gets diverted via the Parlay gateway (2) to the bearer connectivity service in the service domain (3). The proposed approach to connectivity setup is shown in figure 5.

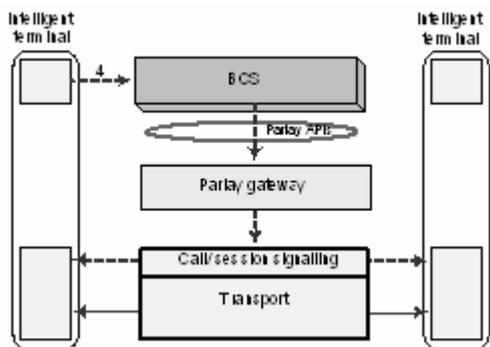


Figure 5: ‘3rd party’ approach to connectivity creation

Using application layer signalling, only a single, direct signal, from the originating terminal to the bearer connectivity service in the service domain, is required (4). This approach is shown below.

This approach can be thought of as using 3<sup>rd</sup> party connection creation to establish a 1<sup>st</sup> party connection. Using application layer signalling to establish a bearer connection is a more efficient approach.

## IV. THE SERVICE DOMAIN ARCHITECTURE

In the previous section, an overview of the principal aspects of the proposed service provisioning environment was presented. Section III (A) provided only a brief overview of the structure of the proposed service domain architecture; this section presents a more complete description.

### A. Architectural overview

Figure 2 presented a simplified version of the proposed service domain architecture. In this section, the service domain architecture is presented in full detail, and the function all major components are described. Figure 6 shows the proposed service domain architecture, in the context of its operational environment.

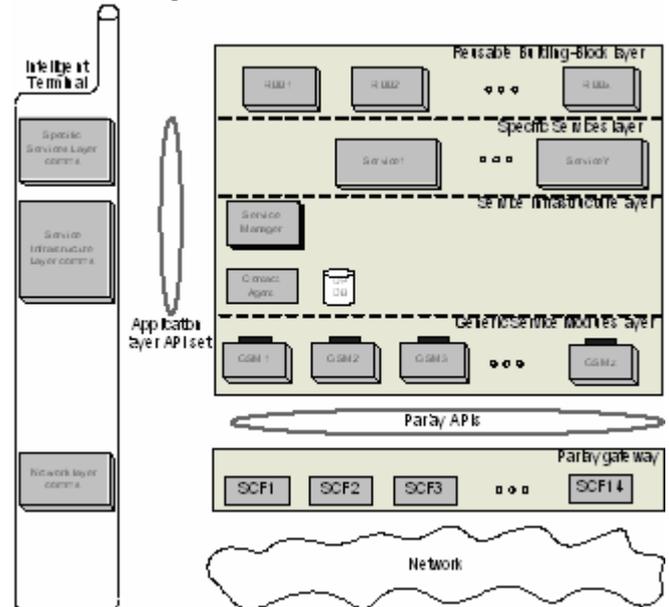


Figure 6: The structured service domain architecture

In the top right of the diagram is the service domain, which is shown to interface with the underlying network via the Parlay gateway, using Parlay APIs. On the left of the diagram, there is a very large intelligent terminal, with functionality spanning all three layers. Within each horizontal layer, elements in the terminal and service domain are able to communicate.

The service domain is divided into 4 horizontal layers, in which the various components of the service domain architecture are found. The second layer from the top, named the *Specific Services layer*, contains the various services. It is ultimately for this layer that the other 3 layers, and the service domain architecture as a whole, exist. The contents of the *Specific Services layer* are provided by service developers.

The uppermost layer is named the *Reusable Building-Block layer*. This layer contains Reusable Building-Blocks (RBBs) that implement commonly required functions. The RBBs allow the services in the *Specific Services layer* to implement only those functions which are unique to that specific service, and ‘outsource’ all common functions to the RBBs.

The *Generic Service Modules layer*, at the bottom of the service domain, contains the Generic Service Modules (GSMs), which are a direct aid to the *Specific Services layer*. The objective of the *Generic Service Modules layer* is to provide an abstraction layer between the Parlay gateway

and the *Specific Services layer*. The contents of the *Generic Service Modules layer* are provided by the operator of the service domain architecture. Whereas the GSMs communicate with the Parlay gateway (on behalf of services), RBBs do not communicate with the gateway, and only interface with the services. The RBBs and the GSMs are discussed in greater depth later in this paper.

Finally, the middle layer, named the *Service Infrastructure layer*, provides the structural framework under which the components of the other 3 layers operate. That is, the *Service Infrastructure layer* in the architecture's nerve centre, and provides the architecture with management and control capabilities. The *Service Infrastructure layer* largely dictates the operational behaviour of the architecture. The contents of the *Service Infrastructure layer* are provided by the operator of the service domain architecture, and are discussed in the following subsection.

### B. Service infrastructure components

The Service Infrastructure layer is comprised of two primary objects: the Service Manager, and the Contact Agent.

#### Service Manager

The Service Manager is a defining feature of the service domain architecture. The Service Manager lies at the heart of the service session, and is the most influential of all the components on the character of the service domain architecture.

A new Service Manager is instantiated for a user at the beginning of every service session, and lasts for only the duration of the service session. The Service Manager is initially instantiated, and ultimately destroyed, by the Contact Agent, described later.

The Service Manager that is established for a particular user for a service session is distinct from other Service Managers. Each and every user has a user/ subscription profile, stored in the *User Profile database*, which contains the details of the services the user is subscribed to. When the Service Manager is instantiated, it is loaded with the user profile of the user, which ensures the user can only invoke services to which he is subscribed.

For both 1<sup>st</sup> and 3<sup>rd</sup> party invocation type services, all service invocation requests are sent to the Service Manager, and never directly to the service. Upon receipt of a service invocation request, the Service Manager determines the appropriate service to be invoked, and subsequently invokes the required service.

Service invocation requests can originate from a number of different technological domains, e.g. the bearer network, web services etc., and invocation requests from each of these domains use a different signalling protocol. Since all service invocation requests are handled by the Service Manager, services are shielded from the technical characteristics of the invoking method, which simplifies the development and implementation of services. The Service Manager provides an adapting function; invocations from different domains, using different protocols, are converted to a common and standard format, which is used for service invocation.

Requiring all service invocation requests to be handled by the Service Manager allows the realisation of a single point

of control for all services. This situation is in contrast with the present approach, where each and every service would be internally managed and operate completely independently of every other service, with no central point of service control and management, and allows the undesirable possibility of service interaction.

#### Contact agent

The Contact Agent plays a very simple, yet essential, role: it instantiates the user's Service Manager at the beginning of every service session, and destroys it at the end.

Outside of the existence of a service session, the only component in the service domain for which a terminal has a reference is the Contact Agent. However, during a service session, the Contact Agent plays no role. The Contact Agent is independent of any user, and is always in existence, ready to be contacted by a terminal to instantiate a Service Manager and initiate a service session.

The Service Manager, the Contact Agent and the user profile database form the *Service Infrastructure layer*. The name of the layer is apt: it is the Contact Agent and the Service Manager that provide the framework for the deployment, invocation and execution of the services, and form the logical infrastructure of the service domain architecture.

### C. Software reuse

There are two primary aids to services shown in figure 5, and discussed here: RBBs, and GSMs. The objective of both of these components is to simplify the development and implementation of services, through reuse and abstraction.

#### Reusable Building-Blocks (RBBs)

RBBs are small building-blocks of commonly required functions and logic that are found to be frequently required by services. Having commonly used software components located in the RBBs allows numerous services to utilise these pre-packaged building-blocks, relieving each service from having to re-implement standard functionality. Instead of the services having to implement all of the methods they require, they are able to make calls to methods already implemented in the RBBs. This renders the services simpler, and through reuse, makes the services more efficient.

RBBs can provide telecoms specific services, such as number translation, or non-telecom services, such as database look-up functions.

#### Generic Service Modules (GSMs)

GSMs serve two purposes. The first purpose of GSMs is to provide an abstraction layer between services and the Parlay gateway, relieving services of having to use the Parlay APIs. The second, much like RBBs, is to enable software reuse by providing services with reusable functions and logic.

Using GSMs, services are now no longer required to communicate with the Parlay gateway directly. Instead, high-level, abstracted method calls can be made by the services on the well-defined north-bound interfaces of the GSMs, which interface with the Parlay gateway. GSMs enable software reuse by encapsulating commonly required Parlay message patterns, used to achieve simple, specific goals, into pre-packaged modules. GSMs reduce the complexity of services, and relieve the service and its

developer from requiring detailed knowledge of the Parlay gateway.

For example, a “Conference Call” GSM would offer services commonly required functions used to establish and control a multimedia conference, and would predominantly interface with the Parlay MPCC SCF.

#### D. The Application layer API set

Section III (B) proposed the use of application layer signalling for communications between terminals and the service domain. The methods offered on the application layer signalling channel are standardised by the Application layer API set, which consists of two categories, corresponding to the direction of the method call:

- Terminals calling methods located on the Contact Agent and the Service Manager
- Services calling methods located on the terminals.

Terminals would call methods located in the Contact Agent for service session initiation, and methods on the Service Manager for service invocation. Services would call methods located in the terminals if additional information is required for successful service execution.

The standardisation of the Application layer API set allows services to be implemented without requiring knowledge of the technological attributes of the underlying network. Thus, the Application layer API set and the Parlay APIs provide similar functions. Both provide the service domain with a network independent view of its operational environment- the underlying network. The difference is that the Parlay APIs standardise the interface between the service domain and the bearer network, whereas the Application layer API set standardises the interface between the service domain and the terminals.

#### V. CONCLUSIONS

The objective of any communications network is to offer its users valuable services. The value of any service is measured by the contribution it makes to the users’ lives, and the contribution a service can make is dependent on its capabilities. Additionally, the existence of value-added services is a function of their ease of implementation, and the number of service developers possessing the skills to develop these services.

The operational environment of the services, within the context of the network, determines both the capabilities and the ease of implementation of services.

The primary objective of this research project was to create an operational environment for services which ultimately creates the most value for the end user. This was attempted through two efforts:

- Design the services’ operational environment so that services can be developed and deployed in the simplest way. This should ensure that the skills required to develop services for communications networks are possessed by the maximum number of service developers.
- Design the services’ operational environment to maximise the capabilities of the network that are offered to the services for utilisation. This would ensure that complex services offering advanced features, as demanded by users, can be provisioned in a simple, standardised way.

These two objectives were attempted to be met by:

- Redesigning the service domain
  - Instituting structure in the service domain, using a layered approach
  - Implementing centralised service control and management
  - Implementing a generic invocation interface for services
  - Advocating software reuse, using RBBs and GSMs
- Modifying the interfaces between the service domain and the rest of the network
  - Relocation of the basic connectivity service control and view to the service domain
  - Introducing application layer signalling

If the ultimate objective of networks is to offer its users a set of value-added services, and services are the focal point of networks’ operations, then the proposed service provisioning environment constitutes a major contribution to the NGN.

#### REFERENCES

- [1] *OSA and Parlay: Enabling an open services market*, Eurescom, <http://www.eurescom.de/message/messageJun2002/tutorial.asp>, Last accessed 24 April 2006.
- [2] Moerdijk A.J., *Parlay/OSA : an open API for service development*, Ericsson. [http://www.parlay.org/en/docs/jan2003mm/27\\_Parlay101.ppt](http://www.parlay.org/en/docs/jan2003mm/27_Parlay101.ppt), Last accessed 24 April 2006.
- [3] Nana P., Mohapi S., Hanrahan H., *Reusable Service Components based in the Parlay API and TINA for the Next Generation Network*, Proceedings of SATNAC 2002, South Africa.
- [4] Farschian B.A., Jakobsson S., Berg E., *Coupling MDA and Parlay to increase reuse in telecommunication application development*, Telenor Research and Development. <http://www.metamodel.com/wiseme-2002/papers/farshchian.pdf>, Last accessed 24 April 2006.
- [5] Schmidt D.C., *Lessons Learned Building OO Telecommunication Software*, Department of Computer Science, Washington University. <http://www.cs.wustl.edu/~schmidt/ACE-lessons.html>, Last accessed 24 April 2006.
- [6] *Parlay Business Values*, The Parlay Group, <http://www.parlay.org/documents>, Last accessed 24 April 2006.

Barry Fricke completed a bachelor’s degree in Electronic Engineering at the University of Pretoria in 2004. He is presently pursuing an MSc (Electrical Engineering) degree at The University of the Witwatersrand.

Hu Hanrahan is a Professor of Communications Engineering at The University of the Witwatersrand. He leads the Centre for Telecommunications Access and Services (CeTAS), a research and advanced teaching centre devoted to improving knowledge and practice in telecoms access and services.