A Meta-Service Creation Environment for the Next-Generation Network (NGN)

Bilal A.R. Jagot, Setumo Mohapi, Hu Hanrahan
Centre for Telecommunications Access and Services
University of the Witwatersrand, Johannesburg
Private Bag 3, Wits, 2050, Gauteng, South Africa
Tel: +27 (11) 717-7229, Fax: +27 (11) 403-1929
Email: {b.jagot, s.mohapi, h.hanrahan}@ee.wits.ac.za

Abstract—We anticipate that the next generation telco will need to develop a large number of services rapidly and cost effectively to gain a competitive edge. The telco will have to choose a Service Creation Environment (SCE) to achieve its goal of rapid and cost effective service creation. Another key goal will be the portability of services between Service Architectures (SAs). Both of these goals can be achieved by adopting the Model Driven Architecture (MDA) of the Object Management Group which will facilitate a platform-independent approach to service creation. A MDA implies that all constructs of the SCE will be represented by a model. By specifying a meta-model that describes the models we achieve portability of services across platforms. The meta-model results in a meta-framework which describes SCEs.

This paper proposes such a meta-framework called Software Engineering based on Modelling for Telecommunications Architectures (SEMTA). SEMTA describes SCEs that have a general platform-independent service creation process that is guided by a methodical telecommunications design process. A SCE built using SEMTA contains a number of viewpoints that are logically linked. This SCE will also contain a number of tools and resources that are defined by a meta-designer to enable the service designer to create services. A key resource is the reusable component called the Facet. Facets were previously defined by the Telecommunications Information Networking Architecture (TINA) Conformance And Testing (CAT) Workgroup for conformance and testing of “TINA” products. Facets are re-defined as reusable components in an implementation language- and platform-independent manner encompassing a particular functionality. This work has been implemented on the SATINA-NGN trial platform.

Index Terms—Next Generation Network (NGN), TINA, Facets, Reusability, Meta-modelling

I. INTRODUCTION

Work has been done on the Next Generation Network (NGN) resulting in a general consensus that the NGN consists of an underlying Network Architecture (NA) and a higher layer Service Architecture (SA) which are interoperable with legacy systems. It is accepted that the NA will be a packet-based, QOS-enabled network that presents a point of abstraction to SAs to enable easier service creation. The main aim of a telco will be to create services in the NGN in an economical manner.

Creating services within a particular SA is well defined but we are presently posed with a number of SAs. Historically, Intelligent Networks (IN) was the first SA to provide services to the consumer. IN was dedicated to using a set of physical components that communicated with each other using different protocols. The service creation process in IN was facilitated by IN’s inherent reusable component called the Service Independent Building Block (SIB). The SIB was script based and built to work with the IN structure. These characteristics proved the SIB to be too rigid, yet, they offered the concept of reusable functional components which Enterprise Java Beans (EJBs) have capitalised on.

EJBs are reusable components, implemented in Java, that are encapsulated within a container and communicate with its environment through a pre-defined interface. Java APIs for Integrated Networks (JAIN) is a SA that takes advantage of the EJB reusability to facilitate service creation by combining multiple EJBs. The disadvantage of EJBs is their implementation language-dependence which would hamper economical service creation within a multi-implementation language environment such as the NGN. A move towards an implementation language and platform-independent reusable component will facilitate economical service creation. Similarly, a service creation process that supports this type of reusable component is required.

Conventional software engineering resources such as Unified Modeling Language (UML) [1] introduce platform-independent service creation but are too general and lack rigour. Methodical telecommunication design processes such as Reference Model for Open Distributed Processing (RM-ODP) [2] will introduce rigour together with direction to the service creation process.

The choice of a SA should not make a difference to the service creation process even though the SAs differ and are rigid in terms of service execution and management. Telecommunications Information Networking Architecture (TINA), for instance, is built on session models and separation of concerns while Parly and JAIN use the conventional Call Model. The service creation process is achieved within a service creation environment (SCE).

The SCE must facilitate portability of services between SAs. Portability is a key issue to enable compatibility of services between SAs. Portability will also prevent unnecessary creation of existing services for new SAs. This SCE should also meet the following requirements:

- Be a model driven architecture (MDA) as specified by the OMG [3],
- abstract underlying protocols from the service designer,
- integrate and interwork with new or legacy infrastructure, operational support systems and services [4],
- enable customisation by partial implementation of components within a particular SA.

This work was supported by a TINA Fellowship award, Telkom SA Limited, Siemens and the THRIP programme of the Department of Trade and Industry
The requirement of a MDA implies that all constructs of the SCE will be represented by a model. Use of a meta-model to describe these models will allow transformation of models from one platform to the next through the meta-model. Hence services described within this type of SCE will be portable across SAs. The meta-model will also result in the creation of a meta-framework that will describe SCEs.

This paper proposes such a meta-framework called Software Engineering based on Modelling for Telecommunications Architectures (SEMTA). SEMTA is a platform independent design that uses models to enable a configurable yet organised design process supported by reusable components. SEMTA empowers two types of developers. The first is the service designer who creates the service from the given tools and processes. The second developer is the meta-designer who designs the tools and processes that a service designer requires. The service designer uses the SCE that is created by the meta-designer. The meta-designer, in turn, uses the meta-framework to design the SCE.

Section II is an overview of SCEs and SEMTA with the objective of explicitly defining the context of this paper. Section III presents a detailed explanation of the Process Framework (PF), which embodies a software engineering lifecycle process. To achieve reusability a model called the Facet is described in Section IV.

II. CONTEXT OF RESEARCH

Understanding of a SCE emerges from the steps during the creation process shown in figure 1. The decision to create a particular service is driven by a number of factors such as a market push or technology pull. Once it has been decided to create service x, with a given specification, a process of resource and environment identification begins. The output of this process is implementation code for service x. The implementation code is created from components and then has to be verified for syntactical and logical errors. Thereafter, the implementation is validated against its original specification. Only after the validation process is completed, can service x be deployed using a service management system or manually.

The proposed solution concentrates on Resource and Environment Identification, the resultant implementation code and support for requirements specification. Resource and Environment Identification is carried out within the Process Framework (PF) of SEMTA shown in figure 2. Models are used in the PF to achieve platform independence, abstraction from underlying technologies and reusability resulting in a SCE that produces services quickly and cost effectively. III expands on the PF.

III. PROCESS FRAMEWORK

The structure of the Process Framework (PF) is the meta-framework whereas the contents of the PF define the SCE. The PF is the environment of the meta-designer and the constituents of the PF are the resources available. The resultant PF (SCE) is the environment of the service designer and the models in the SCE are the resources available to the service designer. The PF is divided into two related parts illustrated in figure 2 namely:

1) **Reference Model (PF-RM)**: this model defines an abstract platform-independent model that explains what needs to be designed. Viewpoints and sub-viewpoints specify the PF-RM. Sub-viewpoints allow the meta-designer to create viewpoints that are more specific. An example of a (PF-RM) is the RM-ODP [2].

2) **Definition Model (PF-DM)**: this model is a mapping and extension of the (PF-RM) to include specifics of SEMTA and the particular platform of choice. The PF-DM inherits the viewpoint and sub-viewpoint structure from the PF-RM. The PF-DM is that part of the PF that contains the various models and is used as the design guide during the service creation process. An example of a PF-DM is the Description Plane Model [5].

The PF-DM introduces a mild automation mechanism by linking models between and within viewpoints. The link between the models, defined in the PF-DM, is defined in the PF Interaction Model (PFIM) illustrated in figure 3. An example of linked models is the creation of a stakeholder in a general business sub-viewpoint resulting in the creation of an actor in a Use Case business sub-viewpoint.

The context of the PFIM includes all constituent models of the PF-DM. Hence only one PFIM may exist in a PF-DM.
The PFIM may contain one or more Viewpoints. The Viewpoint in turn consists of one or more Sub-Viewpoints. Sub-Viewpoints introduce granularity to the level of abstraction defined by the Viewpoint. Sub-Viewpoints are context-specific. Thus, a Sub-Viewpoint cannot be part of multiple Viewpoints. Sub-viewpoints are further composed of Canvasses. Canvasses are the whiteboards where Process Models (PMs), contained in a Tool, are arranged in a logical manner. These constituents are discussed further in the next three sub-sections.

A. Process Models

The Process Model PM provides the service designer with a means of representing ideas. The PM is defined, within a context, by its representation and its User defined aspects, such as textboxes. PMs may exist in multiple Tools but cannot violate the context of the Tool. PMs are created from PM Types (PMTs) and the instances of these PMs are called PM Objects (PMOs). PMOs are placed on Canvasses. Specializing one of seven PMTs creates PMs. The seven PMTs are:

1) **Object**: this PMT represents an object within a given context. An example of a PMO is a Stakeholder in a business model.

2) **Association**: this PMT represents the association between one or more PMs that are of the Object type. An example of a PMO is an association that links a Consumer Stakeholder to a Retailer Stakeholder with a purpose to define the business relationship between the two.

3) **Container**: this PMT represents the containment of a number of instances of PMs. The PMs do not have to be created from the same PMT. An example of a PMO is a Retailer Domain that contains the Service Provider, Connectivity Provider and the associations between them.

4) **Facet**: this PMT represents a reusable component and inherits from the Object PMT. The Facet PMT supports the exploration of the structure and characteristics of a Facet PMO. An example of a PMO is a Stream Binding Facet [6] that creates and manages streams.

5) **Boundary**: this PMT represents boundaries between two or more objects, associations, facets or containers. The boundary separates concerns within a Canvas. An example of a PMO is the boundary between the Consumer Domain and the Retailer Domain.

6) **Comment**: this PMT represents a comment that may be inserted to clarify a particular concept. An example of a PMO is a UML comment box that clarifies or adds more information to a particular aspect of the design.

7) **Miscellaneous**: this PMT represents an arbitrary template, which allows for completeness of PMTs. Numerous PMs can be created to suit individual project needs using the above seven templates.

B. Tools

Tools are a means of allowing PMs to be placed on a Canvas. A Tool is a container for PMs that share a common context with the Tool. The Tool defines its constituent PMs and how they interact with each other. The graphical representation of a Tool is a container whose graphical constructs are the graphical representations of the PMs. Many Canvasses may use a particular Tool but a Canvas may not contain more than one Tool.

C. Canvas

A Canvas graphically captures the service designer’s logical thought as a set of related Process Model Objects (PMOs). The Canvas is defined by the meta-designer and the service designer uses an instance of the Canvas for designing the system. A Canvas can be used by a number of (Sub-) Viewpoints and a (Sub-) Viewpoint can contain a number of Canvasses instances. For example, if two Viewpoints require Flowcharts to be drawn then the same Canvas may be associated with both Viewpoints.

The Canvas is concerned with the placement of PMOs and is defined by five aspects. Figure 4 illustrates the five aspects of the Canvas. The first aspect is the general rules which defines rules that govern the PMOs that are placed on the Canvas. These rules are executed within an algorithm in the Logic aspect. This algorithm will also use the rules that are defined within the Tool. The Tool is incorporated into the Canvas to enable the user to “draw” on the Canvas. The drawing is restricted to three View Types:

1) **Text Only**: this view type supports the inclusion of text only. An example of such a view is the displaying of source code.

2) **Graphics Only**: this view type supports the inclusion of graphics only. Although graphics may contain their own
text fields for descriptive purposes. An example of such a view is a Canvas that displays UML Use Cases.

3) **Multi-media view:** this view has no restrictions with regard to the type or format of information that is placed on the Canvas instance.

The meta-designer may require PMOs to be placed on a Canvas as default whenever an instance of this Canvas is created. The fifth aspect, Default PMs on Canvas (DPC), will record the default instances of the PMs on the Canvas. The DPC works in a manner similar to a macro.

As the development of a service progresses on a set of Canvases, the inclusion of reusable components is necessitated. These reusable components are supported within the structure of a Facet.

### IV. FACET

The TINA Conformance and Testing (CAT) Workgroup defined a facet as a specification template for technical conformance testing of “TINA” products. The context of the facet was inter- and intra-domain reference points where it captured particular functionalities [7]. Hence a reference point is composed of a number of facets enabling a supplier of TINA products to create partial implementations of a reference point. We broaden the definition of Facets for universal reusability in a telecommunications platform (where platform refers to Architectures that contain sub-Architectures as in TINA), yet still encapsulating a particular functionality and enabling partial implementations.

**A Facet is defined here as a multi-granular, platform-independent, implementation-code independent, reusable component that encapsulates a particular functionality and can be tested for conformance to its initial specification at the granularity for which it was designed.**

Multi-granularity is the hierarchical breakdown of software (encompassing functionality) into smaller more manageable components. Figure 5 shows seven levels of granularity. This granularity can extend from a large base such as a Platform down to an Algorithm and may be modified as required as long as **Facets** of one granularity level do not directly interact with **Facets** of a different level. **Facets** display characteristics of inheritance, containment and aggregation. High-level **Facets** contain and use lower level **Facets**. Inheriting from higher-level **Facets** is also possible. The Facet is further explained in section IV-A and an example is given in section IV-B.

#### A. Description of a Facet

The TINA CAT Workgroup presented a well-structured definition of a **Facet** which is also presented here. The definition is extended by the addition of a Technology and Context description. The descriptive terms are as follows:

1) **Context:** specifies the expected area of implementation. Area is a broad term that is used to refer to a business administrative domain, reference point or any other context of application. The Context can be overridden by the service designer at design time.

2) **Interface specifications:** specifies its expected behaviour and interactions using Specification and Description Language (SDL) and Message Sequence Chart (MSC). The overall specifications are written to be independent of the implementation code.

3) **Related facets:** specifies other **Facets** that are associated, directly or indirectly, to enable this **Facet** to accomplish its functionality. This association refers to among others inheritance, containment, and usage.

4) **Technology:** specifies the technology that is used. Technology here refers to the underlying protocols or the specific type of architectural technology.

5) **Usage of the facet:** specifies the intended use of the **Facet** by means of examples. An example of the usage of a database accessing **Facet** is within the TINA Subscriber component to access subscriber information.

This definition of a **Facet** is useful to identify the most suitable **Facets** for a particular function.

The **Facet** supports customisation. A service designer is able to customize a **Facet** to adhere to the naming conventions of the environment in which it is placed. If the **Facet** is composed of other sub-**Facets**, the sub-**Facets** may be substituted with other similar **Facets** at the same granularity level, of similar functional description and the same interface definition.

#### B. An Example of a Facet

An example of a **Facet** that encapsulates functionality at the *Formal Definition Language Level* (FDL) is Named Ac-
cess. Named access is defined using Common Object Request Broker Architecture (CORBA) Interface Definition Language (IDL) and is executed with supporting implementation languages. This Facet allows a known user to authenticate with the Retailer/Service Provider before any service related operations can take place. This facet will be explained in the order of the descriptive terms that were presented in section IV-A.

The context of the Named Access Facet is between Business Administrative Domains and is known as a Reference Point (RP). Figure 6 is an example of this type of context and is known as the Retailer RP. Similarly, the Named Access Facet can be placed within the 3rd Party RP between the Retailer and the 3rd party Service Provider. In this context the Named Access Facet allows the Retailer to be authenticated in the 3rd Party Service Provider’s domain. The movement of this Facet from the Retailer RP to the 3rd Party RP shows reusability of the Named Access Facet.

Figure 6 further shows that the Named Access Facet uses methods from a number of interfaces which are numbered as follows:

1) iProviderInitial
2) iProviderAuthenticate
3) iInitial on User Agent (UA)
4) iProviderNamedAccess

The interface methods are called in a particular sequence shown in a Message Sequence Chart (MSC) to achieve the required functionality. The MSC in [8] illustrates the external behaviour of the Facet as seen by the Provider Agent (PA). The internal behaviour, also illustrated with a MSC or SDL, is exhibited by the server and client implementations for the respective interfaces (1 ... 4). These servers and clients are implemented as object oriented (OO) classes which are Facets at the OO class level. Thus the Named Access Facet at the FDL level uses related Facets at the OO class level.

The Facet uses a number of technologies to achieve communication between components as shown in figure 7. Since the Facet is specified at the FDL level using IDL, the existence of a Distributed Processing Environment (DPE) is implied. The DPE allows objects that are physically separated to communicate with each other as if they existed in the same physical processing environment. This quality of the DPE allows a known user to authenticate with the Retailer/Service Provider. In this context the Facet extends to other Architectures such as the Softswitch Architecture. A JAIN-Parlay network is one such example. In the JAIN-Parlay scenario the Parlay server will authenticate the JAIN network to enable service provision. Although the implementation of the JAIN-Parlay scenario is different from the TINA scenario, the same task is carried out in both situations. The Facet will accommodate these differences by using different lower level Facets.

V. Conclusion

SEMTA, a meta-framework, has been presented to describe SCEs. SEMTA conforms to the MDA and uses metamodeling to enable portability of services between SAs. The meta-framework achieves flexible and rapid service creation through the use of Process Models and reusable components called Facets. Facets have been shown to be implementation language- and platform-independent. By creating a meta-framework, SEMTA empowers a meta-designer to create a SCE and a service designer to create services using this SCE. SEMTA was implemented on the SATINA-NGN trial platform and has to date been tested on small telecommunication service scenarios.

REFERENCES

Bilal A.R. Jagot received his B.Sc. degree in Electrical Engineering at the University of the Witwatersrand, Johannesburg, in 2000. He is a full-time M.Sc. student at the University of the Witwatersrand, where he currently conducts research on modelling for service creation in the Next Generation Network (NGN).

Setumo Mohapi is a Research Officer and PhD Student with the Centre for Telecommunications Access and Services. His current research interests are in the area of management of load in distributed systems. Other fields of interests include distributed computing and mathematical analysis.

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