13

Object Oriented IN service modelling

L. Klostermann, J.A. Kroeze
Ericsson Telecom, the Netherlands
PO Box 8
5120 AA Rijen
the Netherlands
phone +31 161 249911, fax +31 161 249699
e-mail etmlukl@etm.ericsson.se, etmjohk@etm.ericsson.se

Abstract

In this paper we study the possibility to migrate the Intelligent Network Service Model used in standardisation towards an object oriented model. After introducing the current model, we present an initial proposal for an object oriented service model which improves upon the data modelling and service management, while offering migration from the existing model.

Keywords

Intelligent Network, standardisation, service model, object orientation

1 INTRODUCTION

This chapter briefly introduces the existing intelligent network service model.

1.1 Intelligent Networks

Intelligent Networks (IN) started a few years ago, when it became clear that switch based hardcoded services have a disadvantage when it comes to service management. As an alternative an architecture was developed with centralised service control, and a communication protocol between the switching layer and the service layer. In this architecture services are executed in a functional entity called Service Control Function (SCF). An implementation of an SCF is referred to as a Service Control Point (SCP).

1.2 Service Independent Building blocks

The purpose of this centralisation was to improve service management. There was a need for better support of the service life cycle. The concept of Service Independent Building block (SIB) was introduced, aiming at easier and faster service development cycles. In this concept each SIB performs a task occurring regularly in services and services are just a concatenation of SIBs, see figure 1. The basic call process is interrupted when the IN service is triggered, such that the service logic program gains control over the call. From a relatively small set of SIBs many services can be composed, using the flexibility offered by parameters. Each SIB in a service has its associated data, the service has its own data which is partly call related data, so-called call instance data. The flow of service execution is apart from the concatenation of SIBs also determined by incoming events from other functional entities like the Service Switching Point (SSP) or Intelligent Peripheral (IP).

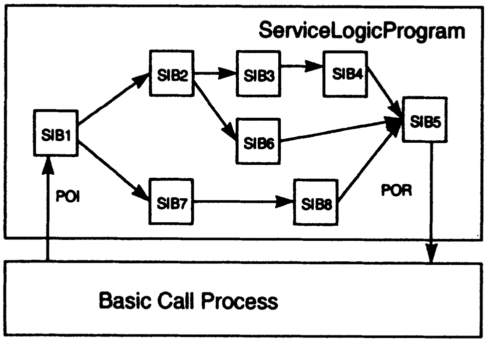

Figure 1 Existing IN service model

1.3 Intelligent network conceptual model

In standardisation of Intelligent Networks, the IN conceptual model is used for protocol development. In this process first some benchmark services are identified. The functionality in these services leads to requirements on the Global Functional Plane (GFP). In turn these requirements may lead to new SIBs.
The functionality of the SIBs is then distributed over the functional entities in the Distributed Functional Plane (DFP), from which a need for a new protocol message between functional entities may be identified.
Finally the protocols are implemented on the physical plane.

2 MOTIVATION FOR A NEW MODEL

This chapter identifies some possibilities for improvement in the existing model.

2.1 Protocol development

With the increased speed of technical developments, growing numbers of protocols and protocol messages are to be supported, having a direct impact on the SIB set. Therefore we see the following improvements related to protocol development:

- from service plane to GFP: The relations between Service Plane and GFP may become more obvious using OO, since the OO model may be derived from a textual specification in a more intuitive way, modelling real-world entities.
- from GFP to DFP: Nowadays there is no methodology to translate the functionality from GFP to DFP. Using OO methodologies supports stepwise refinements, thus distributing functionality at the GFP to functionality over the functional entities of the DFP in a iterative manner.

2.2 Service management

With the current pressure on the service market, service management has again become an issue. Although the GFP service model has currently no direct link to service management and service creation, there are reasons to include it here as well. One such reason is for instance service portability, where a common data model is needed. Possible enhancements we see are:

- Include data modelling aspects in the service model: At present, the context of the SIB modelling is restricted to their functionality in service execution. Data modelling, in particular in the scope of service management, is neglected. Since data modelling is a cornerstone of object orientation, adoptation of OO methodologies will result in a more complete service model.

- Add structure to the SIB set: The set of SIBs presently defined has no structure. On one hand the possibility to make small variations on basic SIBs might be useful, on the other hand a composition of SIBs in units reusable for more services is a valuable addition. The latter aspect is addressed in IN CS2 (Capability Set 2) by so-called high level SIBs. OO inheritance seems a suitable mechanism to support both mechanisms for a SIB set structure.

4 OO SERVICE MODEL

In this chapter we present a proposal for an object oriented service model, which enhances the existing model. We use Object Model Notation (OMT) to represent the models, though we do not always use it in a strictly correct way.

4.1 Service views

The service model supports two views: the execution view and the provisioning/customisation view. The execution view of the service comprises the elementary classes which are active during service execution time, showing classes of different ownership, across levels of subscription and of different temporal character (dynamic or persistent). The provisioning/customisation view on the other hand, represents the perspective of one of the subscribers on the service.

4.2 Service execution view

Meta model

Figure 2 presents the meta model for the service execution view, consisting of the three classes: ServiceSessionManager ServiceLogic and ServiceResource.

The ServiceLogic class represents the flow of control of a service, using the ServiceResource class which models the (physical or logical) resources available in the network for IN services. The ServiceResource class is service independent as opposed to the service specific ServiceLogic class. The ServiceResource class may be looked upon as the OO equivalent of the presently used SIBs.

The ServiceResource class is specialised to the DynamicServiceResource class and the PersistentServiceResource class. The DynamicServiceResource class represents classes of which the instances exist only within the timeframe of one service execution. The PersistentServiceResource class represents the classes of which the instances lifetimes exceed one service execution. Although object lifetime seems a strange criterion for modelling, it separates the objects under control of service management (PersistentServiceResource) from those who are not controlled from service management (DynamicServiceResource).

The ServiceSessionManager keeps a view of the service session and is responsible for (some of the) scheduling, monitoring and event handling. In principle the ServiceSessionManager could be seen as a specialisation of DynamicServiceResource. Because of its central role however, it is modelled as a separate class. ServiceLogic objects may keep private process data.

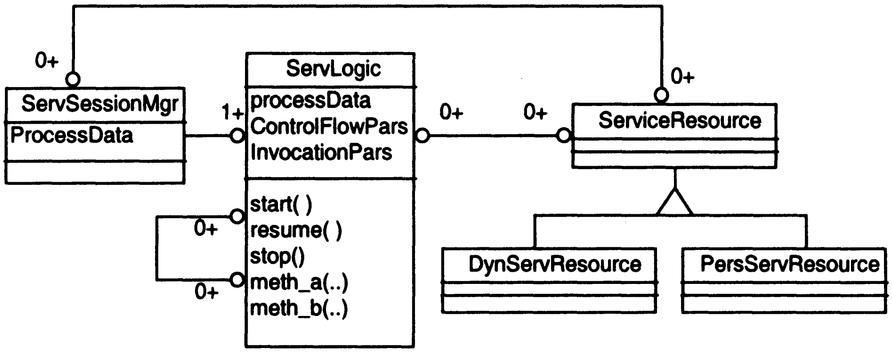

Figure 2 Service execution view meta model

Service creation results in instances of the ServiceLogic class. Service logic processing is started by invoking a method on such an instance. Processing is encapsulated in the ServiceLogic objects. ServiceLogic methods may be invoked synchronous (waiting for the result) or asynchronous (parallel processing).

Refining the meta model

Given the meta model we refine the ServiceResource subclasses. The Call class is a specialisation from DynamicServiceResource, having e.g. Call Instance Data as attributes and methods for call handling.
PersistentServiceResource is specialised to several classes, like announcement, subscriber, queue.

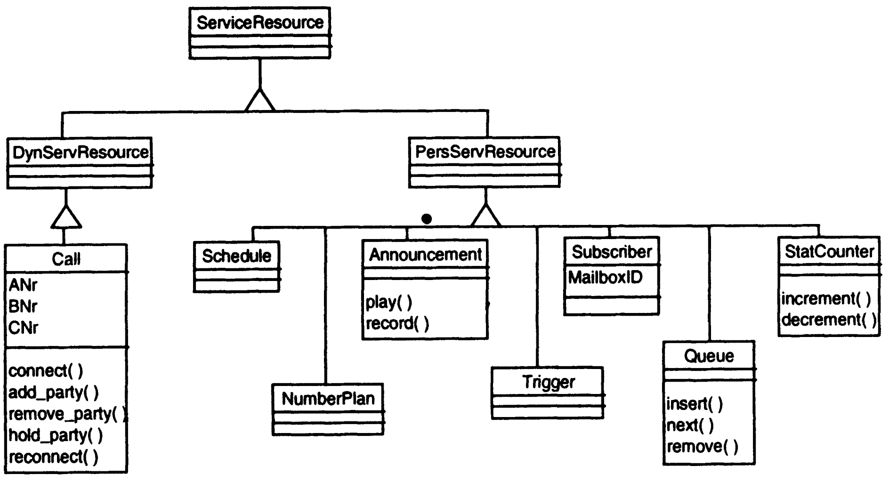

Figure 3 Service execution view, example classes

4.3 Service provisioning/customisation view

The class model of the service provisioning/customisation view is shown in Figure 4. Services are the output of service creation. Services consist of one or more ServiceLogic classes and one or more ServiceTemplate classes.

The ServiceTemplate is used upon subscription, when the subscriber is linked to a service. Subscription may happen on several levels like network provider level, company level, office level or end-user level. For every level there is a template. Once filled out, the ServiceTemplate can be loaded into the network. In this process, several objects are created in the service execution environment. The ProvisioningProfile keeps the bookkeeping of the actual location of the template parameters in the network. During the lifetime of the subscription, the ProvisioningProfile represents the subscription view on the service, for e.g. profile updates by the provider or customer control.

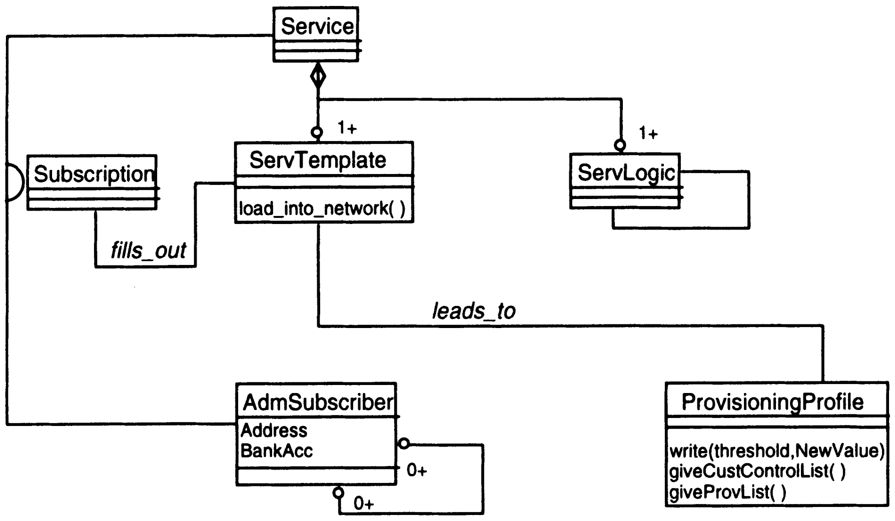

Figure 4 Service provisioning/customisation view

4.4 Combined views

Figure 5 shows the combined service views, illustrating that only the PersistentServiceResource class is linked to the ProvisioningProfile, as opposed to the DynamicServiceResource class.

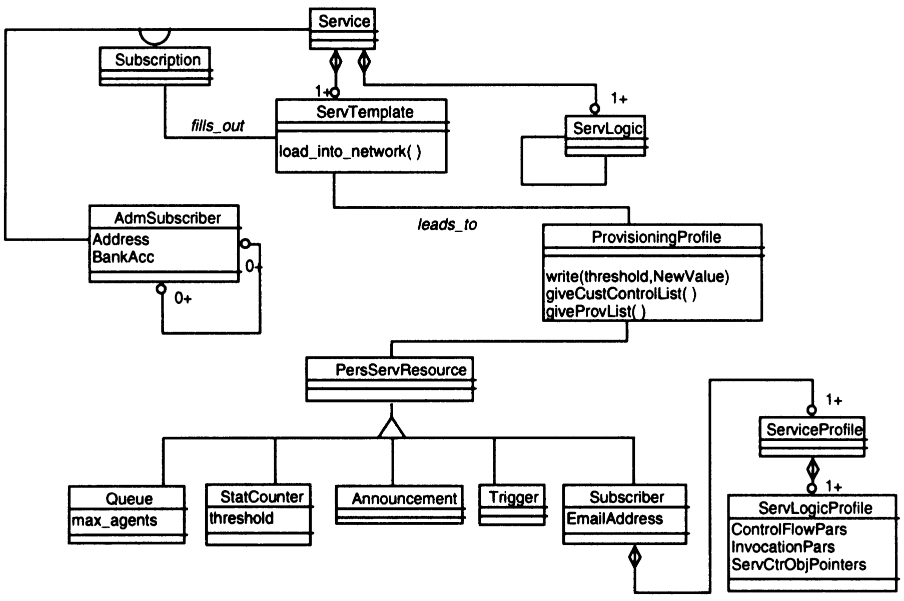

Figure 5 Service combined view

The ServiceProfile and ServiceLogicProfile classes are introduced to represent information which is subscriber specific and thus unknown at service creation time, but still need be linked to the ServiceLogic. In the proposed solution ServiceLogic classes may be reused over many subscribers.

6 CONCLUSION

In this paper we propose an object oriented model for Intelligent Network services. After briefly introducing the current service model we identified two main areas needing improvement, being protocol development in standardisation context, and data modelling, especially important for service management. The proposed model provides improvements in both areas. One of the ways in which this is achieved is by introducing two complementary views. The model is still in a preliminary phase, and further studies are in progress.

7 BIOGRAPHY

Lucas Klostermann (born 1966) obtained a masters degree in physics engineering from Delft University of Technology. After that he worked at CERN in Geneva for four years, doing a PhD in high energy physics. Part of this time he spent developing software for the analysis and interpretation of experimental data from a large experiment. In 1995 he joined Ericsson in the Netherlands, more specifically the IN Application Laboratory of the company. He worked on distributed systems and IN prototyping. Presently he is involved in a TINA validation project and in IN standardisation in ITU.

John Kroeze (born 1965) studied electrical engineering at the University of Twente, resulting in a MSc degree in 1992, after working on his thesis at the Dutch PTT Research laboratory. He worked for three years at the Computing Science Institute of the University of Nijmegen, participating in the European RACE II project R2061 EXPLOIT, where he performed ATM traffic control experiments. In 1995 he joined Ericsson, assigned at the Intelligent Networks Application Laboratory in Rijen, the Netherlands. His current interests include real-time programming, distributed objects, intelligent, broadband and mobile networks (though the intersection of these three seems empty).