Advanced Service Creation: Bridging the Gap Between Requirements Elicitation and Service Design

Dionisis X. Adamopoulos[1], Constantine A. Papandreou[2]

[1] University of Piraeus, Greece and
Centre for Communication Systems Research (CCSR), University of Surrey, UK
[2] University of Piraeus, Greece
Email: {dxa,kospap}@unipi.gr

Abstract. The advent of deregulation combined with new opportunities opened by advances in telecommunications technologies has significantly changed the paradigm of telecommunications services, leading to a dramatic increase in the number and type of services that telecommunication companies can offer. Building new advanced multimedia telecommunications services in a distributed and heterogeneous environment is very difficult, unless there is a methodology to support the entire service development process in a structured and systematic manner, and assist and constrain service designers and developers by setting out goals and providing specific means to achieve these goals. Therefore, in this paper, after a brief presentation of a proposed service creation methodology, its service analysis phase is examined in detail focusing on the essential activities and artifacts. In this process the exploitation of important service engineering techniques and UML modelling principles is especially considered. Finally, alternative and complementary approaches for service analysis are highlighted and a validation attempt is briefly outlined.

1 Introduction

The creation of new advanced telecommunications services (telematic services) within an open network environment with increased intelligence and programmable features is a highly complex activity. This complexity stems not only from the technical nature of the tasks involved, but also from the number of the participating actors and the variety in their roles, concerns, and skills. Therefore, there is a need to support the complex service creation process in order to ensure that resulting services actually perform as planned and as required by customers and service providers. A methodology is an important part of such an attempt, as it provides a systematic and structured base for the flexible and efficient management of the development of telecommunications services.

In this paper, in order to structure and control the service development process from requirements capture and analysis to service implementation, to reduce the inherent complexity, and to ensure the thorough compatibility among the many involved tasks, a service creation methodology, conformant to the open service architectural framework specified by the Telecommunications Information Networking Architecture Consortium (TINA-C) [3, 9], is proposed. Special emphasis is given to

the service analysis phase of the methodology as it provides valuable answers to several important service engineering matters and thus facilitates the transition to a telecommunications environment where many different (enhanced) services are offered by a multiplicity of service providers to several categories of customers within an open market.

2 The Proposed Service Creation Methodology

Telecommunications operators need to master the complexity of service software, because of the highly diversified market demands, and consequently, because of the necessity to quickly and economically develop and introduce a broad range of new services. To achieve such an ambitious, yet strategic to the telecommunications operator's goal, a service creation methodology based on the rich conceptual model of TINA-C is proposed [1].

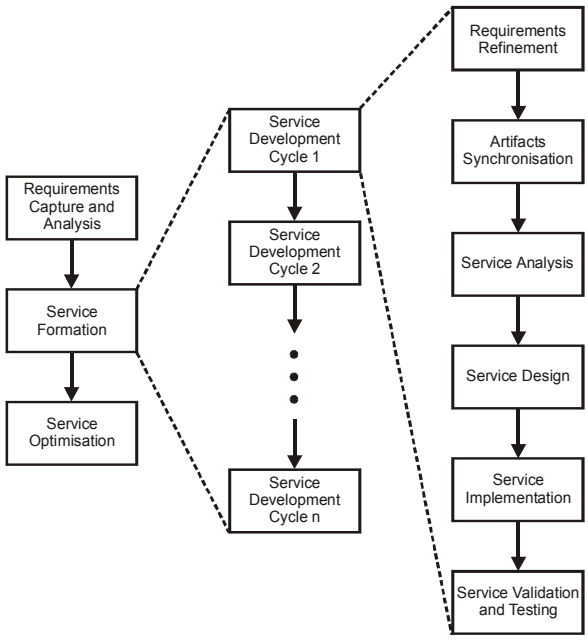

Fig. 1. Overview of the proposed service creation methodology

A high-level or macro-level view of the proposed service creation methodology can be seen in Fig. 1. The proposed service development process is based on an iterative and incremental, use case driven approach. An iterative service creation life cycle is adopted, which is based on successive enlargement and refinement of a telematic service through multiple service development cycles within each one the telematic service grows as it is enriched with new functions. More specifically, after the requirements capture and analysis phase, service development proceeds in a service

formation phase, through a series of service development cycles. Each cycle tackles a relatively small set of service requirements, proceeding through service analysis, service design, service implementation and validation, and service testing. The telematic service grows incrementally as each cycle is completed.

According to Fig. 1 the main phases of the proposed methodology are the following:

- *Requirements capture and analysis phase:* It identifies the telematic service requirements (together with a number of roles), and presents them in a structured way.
- *Service analysis phase:* It describes the semantics of the problem domain that the telematic service is designed for. Thus, it identifies the objects that compose a service (information service objects), their types, and their relationships.
- *Service design phase:* It produces the design specifications of the telematic service under examination. Computational modelling is taking place in this phase and thus the service is described in terms of TINA-C computational objects interacting with each other.
- *Service implementation phase:* In this phase the pieces of the service software (computational objects) are defined and implemented in an object-oriented programming language (e.g. C++, Java), inside a TINA-C compliant Distributed Processing Environment (DPE).
- *Service validation and testing phase:* It subjects the implemented telematic service to a variety of tests in order to ensure its correct and reliable operation.
- *Service optimisation phase:* It examines thoroughly the service code in order to improve its performance in the target DPE, and thus prepare the telematic service for a successful deployment.

As can be seen from Fig. 1, the proposed methodology is conceptually consistent with the viewpoint separation as advocated by TINA-C in accordance with the Reference Model for Open Distributed Processing (RM-ODP). It has to be stressed that the proposed methodology does not imply a waterfall model in which each activity is done once for the entire set of service requirements. Furthermore, graphical and textual notations are proposed for almost all phases to improve the readability of the related results and ensure a level of formalism sufficient to prevent any ambiguity. In the following paragraphs the service analysis phase of the proposed methodology is examined focusing on its essential characteristics and artifacts.

3 The Service Analysis Phase

The aim of this phase is to determine the functionality needed for satisfying the service requirements that were identified in the requirements capture and analysis phase and to define the software architecture of the service implementation. For this reason, the focal point shifts from the service boundary to the internal service structure [1].

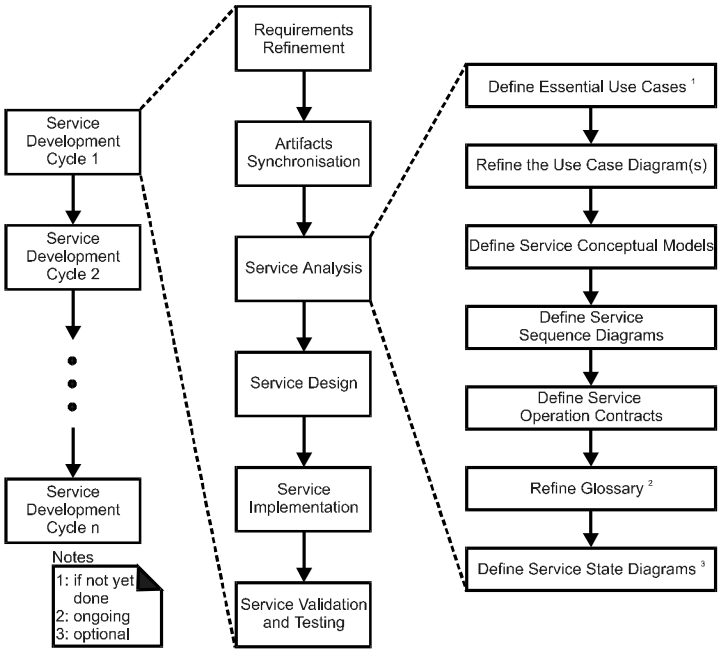

Fig. 2. Service analysis phase activities

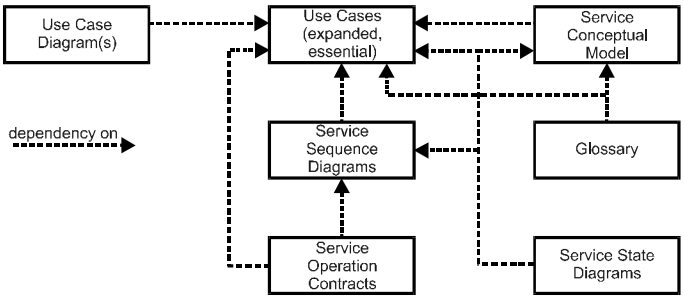

Fig. 3. Service analysis phase artifact dependencies

The activities that take place in this phase can be seen in Fig. 2. The linear order that may be inferred from this figure is not strictly the case, as some artifacts may be created in parallel (e.g. the service conceptual model and the glossary). The dependencies between the artifacts produced can be seen in Fig. 3. The most important activities of this phase are examined in the following sections.

3.1 Definition of Service Conceptual Models

The service analysis phase is the first phase of the service creation process where the telematic service is decomposed into its constituent parts (service information objects

or service concepts), with the appropriate relationships among them, in an attempt to gain an overall understanding of the service. The resulting (main) *service conceptual model*, which is the most important artifact that is created during the service analysis phase, represents a restatement, in a graphical notation, of the problem statement, as it was expressed in the previous phase.

It involves identifying a rich set of *service concepts* regarding the telematic service under examination by investigating the service domain and by analysing the essential use cases. Therefore, it describes what the service is in terms of interesting and meaningful (to the service developer) entities / concepts that constitute it and couplings / associations between them [4, 6]. These couplings define relationships between two *service Information Object (IO)* classes. In UML, a service conceptual model can be illustrated with a set of static structure diagrams in which no operations are defined. It has to be stressed that a service conceptual model is a representation of real-world concepts or actual things, and not a representation of software components (software entities). However, there is no such thing as a single correct service conceptual model. All service conceptual models are approximations of the service domain under examination [8].

The main service conceptual model is accompanied by a set of *ancillary service conceptual models*. These models are derived by (and correspond to) a number of generic information models deduced from the TINA-C service architecture [9] and complement semantically the main service conceptual model with useful session related concepts and structures. More specifically, the ancillary models refer to the modelling of session roles, (TINA-C) sessions, access and service sessions, and to the classification of access and service sessions.

This activity of the service analysis phase consists mainly of the following steps:
Step 1: Identify the service concepts.
Step 2: Identify associations between the service concepts.
Step 3: Identify attributes of the service concepts.
Step 4: Draw the main service conceptual model.
Step 5: Specify the ancillary service conceptual models.

These steps are examined in detail in the following paragraphs grouped into two subsections: one related to the main service conceptual model (steps 1-4), and one related to the ancillary service conceptual models (step 5).

3.1.1 Specification of the Main Service Conceptual Model

The main service conceptual model is specified by the following steps:

Step 1: Identify the service concepts.

A central task when creating a service conceptual model is the identification of the service concepts. It has to be noted as a general guideline that it is better to overspecify a service conceptual model with many fine-grained concepts than to underspecify it. However, it is possible that some service concepts will be missed during the initial attempt to identify service concepts. These concepts will be discovered later during the consideration of associations or attributes, or during the service design phase. When found, the initially missed service concepts, should be added to the service conceptual model.

Two techniques are proposed for the identification of service concepts. The first is based on the use of a *service concept category list*, which contains categories that are usually worth considering, though not in any particular order of importance. Another useful technique is to consider the noun phrases in the text of the expanded use cases as candidate service concepts or attributes. However, this *noun phrase identification technique* must be carefully applied, as a mechanical noun-to-concept mapping isn't possible, and words in natural languages are ambiguous. Nevertheless, this technique is recommended to be used in combination with the service concept category list technique.

Taking into account the previous discussion, the following actions take place when identifying the service concepts during this step:

- Apply the service concept category list technique.
- Apply the noun phrase identification technique.
- Draw an initial service conceptual model by representing graphically only the service concepts.
- Identify useful type hierarchies (optionally).

Step 2: Identify associations between the service concepts.

After identifying the service concepts, it is also necessary to identify those associations of the service concepts that are needed to satisfy the information requirements of the current use case(s) under development and which aid the comprehension of the service conceptual model. The associations that should be considered in order to be included in a service conceptual model are the associations for which the service requirements suggest or imply that knowledge of the relationship that they present needs to be preserved for some duration ("need-to-know" associations) or are otherwise strongly suggested in the service developer's perception of the problem domain. Furthermore, it is usually worth considering, a *common associations list* which contains some common categories of associations.

It has to be noted that it is generally undesirable to overwhelm the service conceptual model with associations that are derivable, not strongly required and which do not increase understanding. Too many associations tend to confuse a service conceptual model rather than clarify it. Their discovery can be time-consuming and with marginal benefit. However, associations should also ensure that an essential understanding of the important service concepts is offered by the service conceptual model.

Taking into account the previous discussion, the following actions take place when identifying associations between the service concepts during this step:

- Find the need-to-know associations.
- Consider the common associations list.
- Consider aggregation (composite or shared), derived, qualified, and recursive or reflexive associations, based on their value in improving understanding of the service domain (optional action).
- Select the desirable associations that will be included in the main service conceptual model.

Step 3: Identify attributes of the service concepts.

A service conceptual model should include all the attributes of the identified service concepts for which the service requirements suggest or imply a need to remember information. These attributes should preferably be *simple attributes* or *pure data values*. Caution is needed to avoid modelling a (complex) service concept as an attribute or relating two service concepts with an attribute instead of an association.

Step 4: Draw the main service conceptual model.

The main service conceptual model is formed by adding the identified type hierarchies, associations and attributes to the initial service conceptual model. It has to be noted that a verb phrase should be used for naming an association, in such a way that the association's name together with the names of the service concepts that it relates create a sequence that is readable and meaningful. Finally, associative types, which are related to an association between two service concepts, should be added to the service conceptual model if considered to be necessary. Usually, associative types are useful when two service concepts are linked with a many-to-many association and there is important information related to it.

3.1.2 Specification of the Ancillary Service Conceptual Models

The proposed methodology considers the TINA-C service architecture in a critical manner with the intention to extract from it useful concepts and guidelines / techniques. Taking into account this attitude, a number of generic TINA-C information models were formed, by considering parts of the documentation that is available by TINA-C [9]. The customisation of these information models according to the service requirements results in the ancillary service conceptual models, which in combination with the main service conceptual model, describe the semantics of the service domain under examination in a clear, concise, and unambiguous way.

The *generic TINA-C information models* that are the basis of the ancillary service conceptual models are briefly described in the following paragraphs. It is stressed that these information models have a pure conceptual / analytical role and they are considered by the proposed methodology released from most of the details that the TINA-C service architecture associates with them (e.g. feature sets).

The session related information models

The session information model is depicted in Fig. 4(a) and is valid for all session related telematic services. As a session related service may extend over multiple business administrative domains, it is intuitive and useful to model the service as an aggregation of one or more domain services, where each domain service represents a part of a service confined to a single domain. Just as a session represents an instance of a service, a domain session represents an instance of a domain service. Domain sessions may interact to establish services extending over multiple domains.

As can be seen in the session role information model, which is depicted in Fig. 4(b), business administrative domains are able to take different session roles during the access and usage interactions that occur between them.

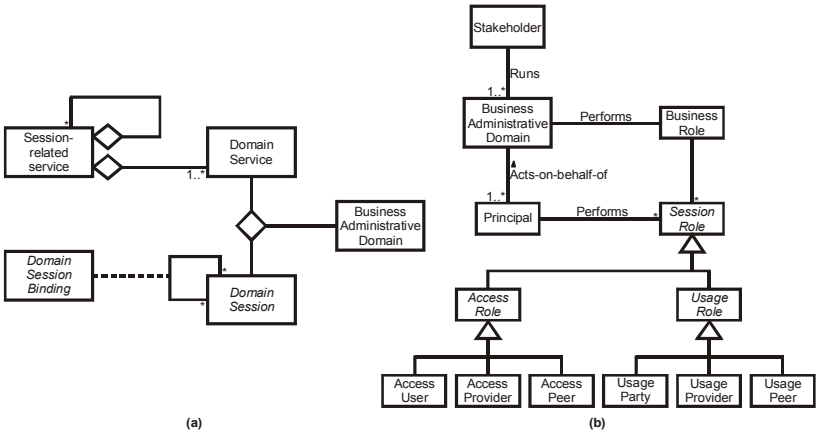

Fig. 4. Important session related information models:
(a) The session information model, (b) The session role information model

The access session related information models

Taking into account the generic TINA-C session related information models of Fig. 4 and the different types of sessions that can be established between business administrative domains, access sessions can be classified according to the specialisation hierarchy shown in Fig. 5(a).

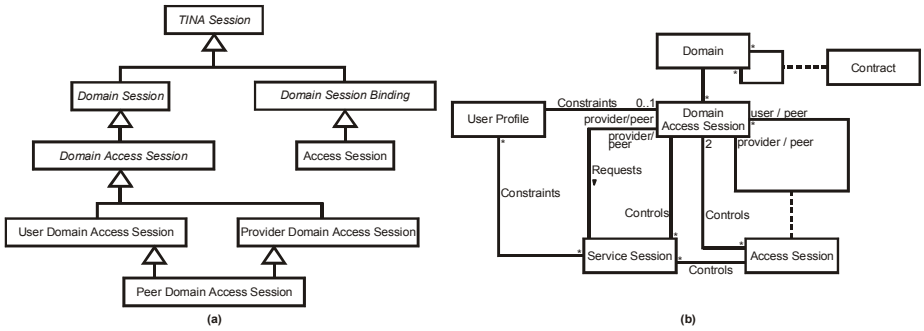

Fig. 5. Important access session related information models:
(a) Classification of the access session, (b) The access session information model

The access session related service IOs and their relationships are depicted in the information model of Fig. 5(b). In this figure, the Domain Access Session (D_AS) service IO is associated with a particular domain and represents the generic information required to establish and support access interactions between two domains. Furthermore, it is specialised into UD_AS (managed by the user), PD_AS (managed by the provider) and PeerD_AS service IOs, as each D_AS is associated with a particular access role. All information that is used directly by the D_AS for authorisation decisions, constraints and customisation of the D_ASs, Access Sessions and Service Sessions is contained in the User Profile.

The service session related information models

Service sessions can be classified according to the specialisation hierarchy shown in Fig. 6(a). The service session related service IOs and their relationships are depicted in the information model of Fig. 6(b). Every service session consists of usage and provider service sessions. Each member of a session, i.e. an end-user, a resource or another session, is associated with a usage service session. Furthermore, each usage service session can extend over two domains and is composed of two complementary Domain Usage Service Sessions (D_USSs). The Domain Usage Service Session Binding (D_USS Binding) represents the dynamic information associated with the binding of two D_USSs.

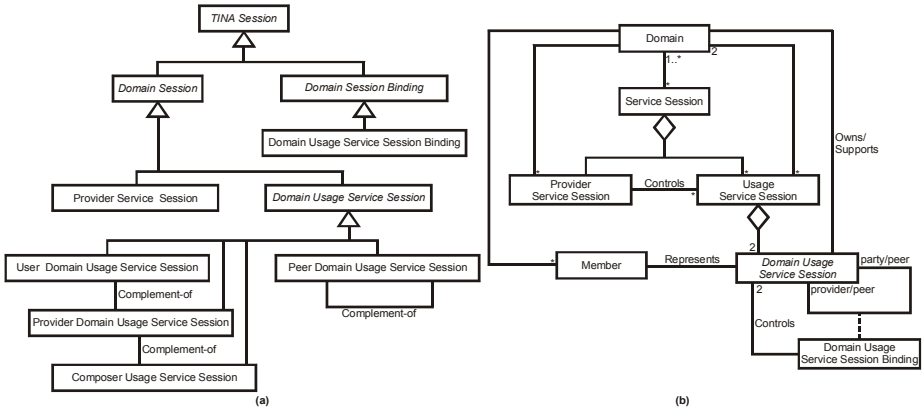

Fig. 6. Important service session related information models:
(a) Classification of the service session, (b) The service session information model

The Service Session Graph information model

The Service Session Graph (SSG) offers a generic framework to describe information in service sessions and is used to model and control the state of a service session. The capabilities modelled in the SSG, which can be seen in Fig. 7, are party invitation and addition, stream binding and stream composition, and explicit control of the use of resources (e.g. various devices participating in continuous media communication). The SSG supports the definition of control relationships through the ControlSR IO type. Additionally, the SSG supports the definition of composing relationships through the CompSR IO type, which facilitates the representation of composition and federation of services. Finally, stream bindings are modelled by the association of StreamInterface IOs to StreamBindingSR IOs via the appropriate SessionMember IOs. The StreamFlowEndPoint IOs have associated Quality of Service (QoS) attributes and can be aggregated into stream interfaces at a party's end system or at a Resource IO.

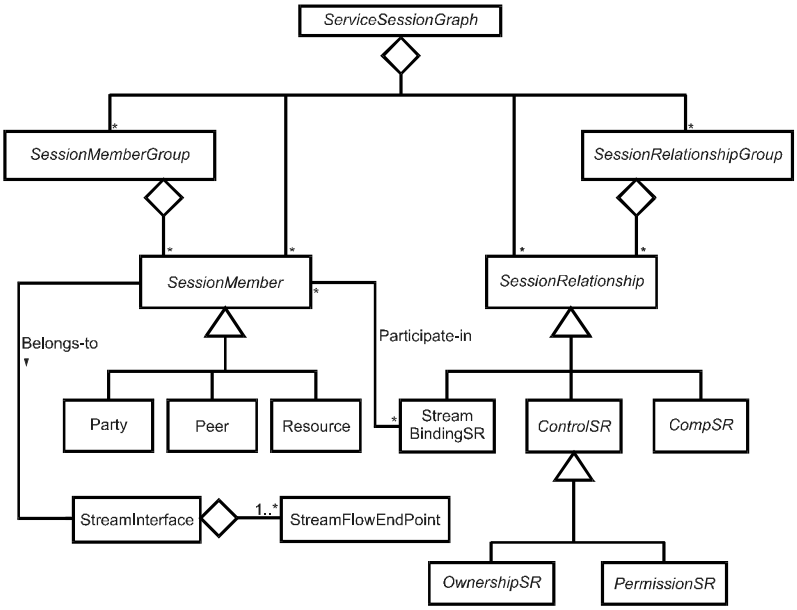

Fig. 7. The Service Session Graph information model

Taking into account the previous discussion, the following actions take place when specifying the ancillary service conceptual models:

- Customise the session related information models (Fig. 4) according to the service requirements.
- Customise the access session related information models (Fig. 5) according to the service requirements.
- Customise the service session related information models (Fig. 6) according to the service requirements.
- Customise the SSG information model (Fig. 7) according to the service requirements.

3.2 Definition of Service Sequence Diagrams

Before proceeding to a logical design of how a telematic service will work in terms of software components, its behaviour is necessary to be examined and defined as a black box. In this way, *service behaviour* is considered as a description of what the telematic service does, without explaining how it does it. One part of that description is service sequence diagrams.

A service sequence diagram should be done for the typical course of events of each use case and sometimes for the most important alternative courses. It depicts, for a particular course of events within a use case, the external actors that interact directly with the telematic service, the telematic service (as a black box), and the service

events that the actors generate. Time proceeds downwards and the ordering of events should follow their order in the use case. Service events (and their associated service operations) should be expressed in an abstract way, emphasising their intention, and not in an implementation specific manner.

Taking into account the above mentioned guidelines and remarks, this activity consists mainly of the following steps:

Step 1: Draw a vertical line representing the telematic service as a black box.
Step 2: Identify each actor that directly operates on (or interacts with) the telematic service.
Step 3: Draw a vertical line for each actor.
Step 4: Identify the (external) service events that each actor generates by examining the use case typical course of events text.
Step 5: Illustrate the identified service events in the correct order on the diagram.
Step 6: Include (fragments of) the use case text to the left of the diagram (optionally).

3.3 Definition of Service Operation Contracts

The behaviour of a telematic service is further defined by service operation contracts (or service contracts), as they describe the effect of service operations upon the telematic service. A service sequence diagram depicts the external events that an actor generates, but it does not elaborate on the details of the functionality associated with the service operations invoked. All the details that are necessary to understand the service response (and thus the actual service behaviour) are missing. These details are included in service operation contracts, which describe changes in the state of the overall telematic service when a service operation is invoked.

The creation of *service operation contracts* is dependent on the prior development of use cases and service sequence diagrams, and on the identification of service operations (see Fig. 3) in the following order:

use cases → service sequence diagrams → service operations
→ service operation contracts

More specifically, the use cases suggest the service events and lead to the construction of the service sequence diagrams. The service operations can then be identified. The effect of the service operations is described in service operation contracts. UML contains support for defining service contracts by allowing the definition of pre- and post-conditions of service operations [5].

A service operation contract is proposed to include the following sections:

Name:	Name of the service operation and its parameters.
Responsibilities:	An informal description of the purpose of the service operation.
Type:	Name of type that the service operation belongs to.
Cross References:	Pointers to important related artifacts (e.g. service function reference numbers, names of use cases, etc.).
Notes:	Design statements regarding the service operation (design guidelines, algorithms, etc.).
Exceptions:	Reaction of the service operation to exceptional situations.
Output:	Non user interface outputs.
Pre-conditions:	Assumptions about the state of the telematic service before execution of the service operation.
Post-conditions:	The state of the telematic service after completion of the service operation.

The most important section is the "Post-conditions" section which declaratively describes the state changes that occur to service IOs in the service conceptual model, using a number of suitably selected statements (instance creation, instance deletion, attribute modification, association formed, association broken, and user interface activation). This section reveals how the telematic service has changed as a result of a specific service operation.

3.4 Definition of Service State Diagrams

The legal sequence of external service events that are recognised and handled by a telematic service in the context of a specific use case can be successfully described by service state diagrams. These are UML state diagrams, which illustrate the interesting and significant service events and the states of a telematic service, together with the behaviour of the service in reaction to a particular service event.

A service state diagram which depicts the (overall) service events and their desired sequence within a use case is called a *use case service state diagram*, and can be created for a specific use case at varying levels of detail depending on the exact modelling needs. The real value of use case service state diagrams is appreciated when they model complex use cases with many service events, because then they help considerably the service developer(s) during the service design to avoid out-of-sequence service events and the corresponding error conditions. However, use case service state diagrams are not necessary if there is no significant service event ordering. Therefore, their definition in the service analysis phase is optional. In such cases, another (optional again) alternative is the creation of a *global service state diagram*, which illustrates, for the entire telematic service, all the transitions for service events across all the use cases. It is a union of all the use case service state diagrams and is useful as long as the total number of service events is small enough to keep the diagram comprehensible.

Taking into account the above discussion, this activity consists mainly of the following steps:

Step 1: Draw a service state diagram for each use case or (alternatively) draw a global service state diagram.

Step 2: Add to the diagram(s) transition actions, transition guard conditions, and nested states (optionally).

3.5 Complementary and Alternative Approaches

In the service analysis phase, high level Object Modelling Technique (OMT) diagrams can be used to represent the service concepts and their relationships. Additionally, analysis level Message Sequence Chart (MSC) diagrams can model the interaction among service parts. However, both these OMT and MSC models provide a high level overview and an understanding of the service as a whole. They do not focus to the individual service IOs. For this purpose, quasi GDMO (Guidelines for the Definition of Managed Objects) and GRM (General Relationship Model) can be used, with quasi GDMO describing the characteristics of service IOs and GRM specifying the

relationships among service IOs. Quasi GDMO and GRM are used for information modelling purposes in TINA-C and originate from the Open Systems Interconnection (OSI) GDMO and GRM formal techniques that were developed for defining managed objects and their relationships. Finally, for the identification of service concepts and service objects, for the assignment of responsibilities to service classes, and for finding and examining collaborations between service classes, CRC (Class - Responsibility - Collaborator) cards can be used. This simple but effective technique, assumes the existence of written requirements specifications and is usable at both the analysis and the design levels.

4 Concluding Remarks

The activities of the service analysis phase can be seen in Fig. 2 and the artifacts that are produced in Fig. 3. Service sequence diagrams and service operation contracts are part of the service behaviour model of the *service analysis model* (see Fig. 8). The service behaviour model specifies what service events a telematic service responds to, and what responsibilities and post-conditions the corresponding service operations have. Furthermore, it describes the external interface and behaviour of the overall service. It has to be noted that in order to be complete (and really object-oriented) the information specification of the service should also take into account the (dynamic) behaviour of individual service IOs. This behaviour is usually defined by allocating operations to the service IOs. However, the issue whether operations should be ascribed to individual service IOs is quite controversial as it actually represents a functional decomposition of the overall service functionality [4, 7]. This clearly implies design decisions that should be better taken at the service design phase.

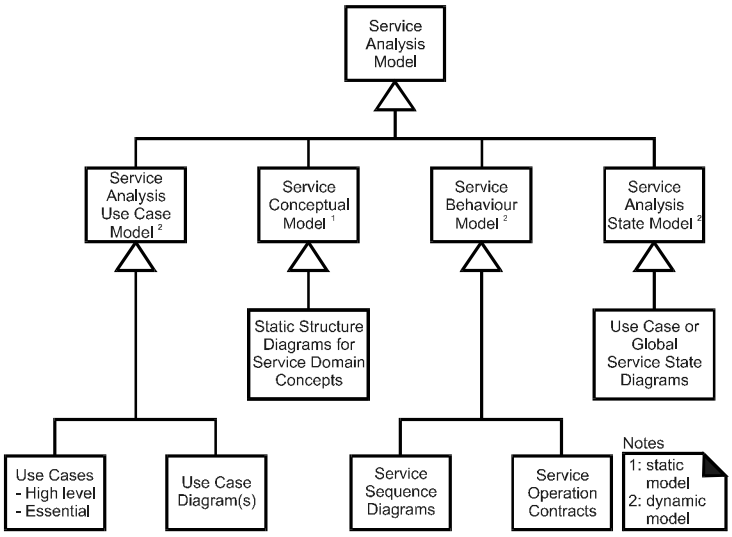

Fig. 8. The service analysis model

Finally, it has to be stressed that the proposed service creation methodology (and thus its service design phase) was validated and its true practical value and applicability was ensured as it was applied to the design and development of a real complex representative telematic service (a MultiMedia Conferencing Service for Education and Training, MMCS-ET). More specifically, a variety of scenarios were considered involving the support of session management requirements (session establishment, modification, suspension, resumption, and shutdown), interaction requirements (audio / video, text, and file communication), and collaboration support requirements (chat facility, file exchange facility, and voting). Considering all the artifacts produced in the service design phase, the MMCS-ET was implemented using Microsoft's Visual C++ together with Microsoft's Distributed Component Object Model (DCOM) [2] (appropriately extended with a high-level API in order to support continuous media interactions) as a distributed object-oriented environment.

References

1. Adamopoulos D.X., Pavlou G., Papandreou C.A.: Advanced Service Creation Using Distributed Object Technology. *IEEE Communications Mag*azine 40(3):146-154 (2002)
2. Adamopoulos D.X., Pavlou G., Papandreou C.A.: Continuous Media Support in the Distributed Component Object Model. *Computer Comm*unications 25(2):169-182 (2002)
3. Berndt H., Hamada T., Graubmann P.: TINA: Its Achievements and its Future Directions. *IEEE Communications Surveys & Tutorials* 3(1) (2000)
4. Declan M.: Adopting Object Oriented Analysis for Telecommunications Systems Development. *Proceedings of IS&N Conference,* Springer LNCS Vol.1238, (1997) 117-125
5. Evits P.: *A UML Pattern Language*. Macmillan Technology Series (2000)
6. Fowler M.: *Analysis Patterns: Reusable Object Models*. Addison Wesley (1996)
7. Gaspoz J.P.: Methodology for the Development of Distributed Telecommunications Services. *Journal of Systems and Software* 3(3)253-275 (1996)
8. Larman C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall (2002)
9. TINA-C: Service Architecture, Ver. 5.0 (1997).