A UML Based Methodology for the Creation of TINA Compatible Telecommunications Services

Dionisis X. Adamopoulos, George Pavlou
Centre for Communication Systems Research
University of Surrey, Guildford, GU2 7XH
{D.Adamopoulos, G.Pavlou}@eim.surrey.ac.uk

Constantine A. Papandreou
Hellenic Telecommunications Organisation (OTE)
17 Kalliga Street, GR-114 73, Athens, Greece
kospap@org.ote.gr

Abstract

In the emerging deregulated multi-provider telecommunications market place, telecommunications service engineering, enhanced with the architectural framework of the Telecommunications Information Networking Architecture Consortium (TINA-C), promises to pave the way towards an open integrated broadband service infrastructure. In order to be able to populate this infrastructure with a variety of new (sophisticated) telecommunications services, it is necessary to assist and constrain service designers during the complex process of service creation. Therefore, in this paper, after considering briefly a number of service modelling issues, a TINA-C conformant service creation methodology, based on the exploitation of the flexible Unified Modelling Language (UML) notation, is proposed. This methodology is then presented focusing on the role and purpose of essential artifacts, examining their dependencies and illustrating the significance of UML.

1. Introduction

The telecommunications market is currently characterised by a fast evolving technology, a multiplicity of cooperating and competing providers, and a growing demand for a great variety of new specialised multimedia services. Recent developments in broadband networks and distributed computing have increased the feasibility of creating telecommunications services as distributed application programs in open distributed systems. These systems are characterised by the heterogeneity of the underlying computing and networking technology. In order to deal with this heterogeneity many distributed platforms (generically called Distributed Processing Environments, DPEs) have been developed [2].

To meet the needs of future telecommunications in an integrated and effective manner, various architectural frameworks are being developed and applied (TINA-C, OSA), that provide the means to build services and a service support environment by attempting to unify and evolve the technologies of Intelligent Networks (INs) and Telecommunication Management Network (TMN). One of the major goals of these architectural frameworks is to assist and constrain designers in the complex process of service creation and design [1][5].

This paper considers important issues that underpin the service creation process in a highly competitive environment of service provisioning. For this reason, a TINA-C conformant service creation methodology is proposed with the intention to accelerate the service life-cycle so that new and enhanced services (telematic services) can be developed and deployed at a faster rate, in a cost effective manner, without making quality compromises in an open deregulated multi-provider telecommunications market place.

2. The modelling approach

A telematic service, due to its (potentially) enhanced functionality and its inherent distributed nature, is usually overwhelmingly complex and thus it is necessary to decompose it into understandable parts / segments in order to fully comprehend its semantics and manage the complexity. These parts / segments may be represented as models which describe and abstract essential aspects of the telematic service.

Therefore, a useful activity during the development of a telematic service is to create models of the service, which organise in a concise way and communicate with accuracy the important details of the telematic service under examination. These service models should contain cohesive, strongly related elements and are usually composed of other (simpler) models or artifacts, comprising basically diagrams and documents which describe concepts and entities, and reveal the relations between them. Service models in the proposed methodology are specified using the Unified Modelling Language (UML), which is an emerging industry standard for specifying, constructing, visualising, and documenting software-intensive systems [3].

Finally, it has to be stressed that the overall modelling approach followed by the proposed methodology is strongly influenced by the TINA-C service architecture and by the modelling guidelines and rules that it encom-

0-7695-0722-0/00 $10.00 © 2000 IEEE

passes [7]. Services are considered as software-based applications that operate on a distributed computing platform which supports the portability and interoperability of the service code. Therefore, a telematic service is realised by a set of interacting service components, which are distributed across different network elements.

3. 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 [4]. To achieve such an ambitious, yet strategic to the telecommunications operators goal,a service creation methodology based on the rich conceptual model of TINA-C is proposed.

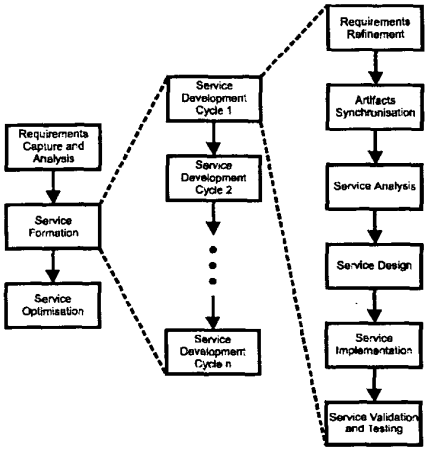

Figure 1. Iterative service development cycles in the proposed methodology

A high-level or macro-level view of the proposed service creation methodology can be seen in Figure 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. The proposed use of the UML notation [3], in almost all the phases of the methodology, has the advantage of making both the service description more coherent and the process of proceeding from one phase to another more natural and efficient.

As can be seen from Figure 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) and 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 most important phases of the proposed methodology are examined focusing on their essential characteristics and artifacts. The service optimisation phase has been omitted, as it depends highly on the selected programming language and on the target DPE.

3.1. Requirements capture and analysis

During this phase the service developer assembles, documents, and structures the requirements on the service (service needs) from the different stakeholders involved. The focus is on modelling the concepts that are visible at the service boundary, and thus the service logic is viewed as a black box.

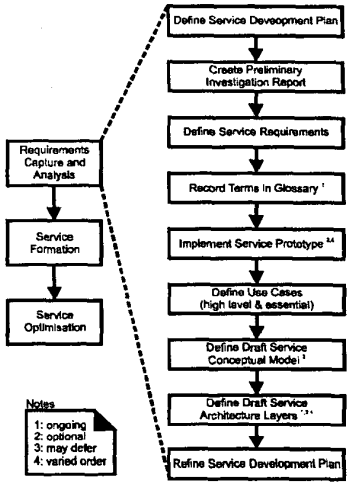

Figure 2. Requirements capture and analysis phase activities

The activities that take place in this phase can be seen in Figure 2. Although this figure suggests a linear order of artifact creation, that is not strictly the case. Some artifacts may be created in parallel. This is especially true for the draft conceptual model, the glossary, the use cases, and the use case diagram(s). The dependencies between the artifacts produced during the requirements capture and analysis phase can be seen in Figure 3.

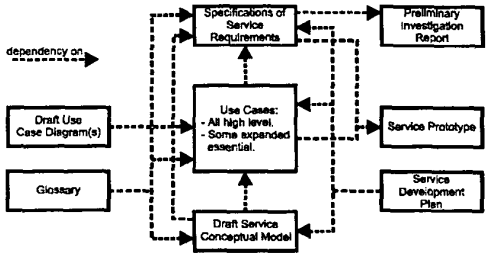

Figure 3. Requirements capture and analysis phase artifact dependencies

The service developer after gathering enough material regarding the telematic service under examination with various means (e.g. interviews, group meetings, study of related documents, etc.) and in various forms / formats (e.g. notes, sketches, audio recordings, etc.), attempts to process this material and structure it appropriately in order to elicit from it a set of service requirements.

In full agreement, one of the most important tasks that the service developer has to perform is the identification of the independent entities / actors, which are involved (by their collaboration) in the operation of the service within and across business administrative domain boundaries. These entities correspond to roles modelling a well defined grouping of functionality under control of a specific stakeholder. This initial task is important because the gathered requirements are structured around the identified roles by determining relationships between the roles. A role can be either generic or specific. The main generic (business) roles and the (business) relationships between them are specified by the TINA Business Model [8]. Each generic role (consumer, retailer, broker, third party service provider, connectivity provider) corresponds to one or more specific roles.

In order to improve the understanding of the service requirements, use cases are created. Use cases are textual narrative descriptions of service domain processes. They describe the sequence of events generated by an actor using a telematic service to complete a specific service process. Use cases may be expressed with varying degrees of detail and commitment to design decisions. Therefore, the same use case may be written in different formats, with different levels of detail. There are two basic formats that a use case can take, leading respectively to high-level use cases and to expanded use cases.

A high-level use case describes a service process very briefly, usually in two or three sentences. It is useful to create this type of use case during the requirements capture and analysis phase in order to quickly understand the degree of complexity and functionality in a telematic service. High-level use cases are very terse and vague on design decisions. On the other hand, an expanded use case describes a service process in more detail than a high-level one. The primary difference from a high-level use case is that it has a section which describes the step-by-step events.

Furthermore, use cases can be either essential or real. More specifically, essential use cases are expanded use cases that are expressed in a form that remains relatively free of technology and implementation details, as design decisions (especially those related to the user interface) are deferred and abstracted. It is desirable to create essential use cases during the requirements capture and analysis phase in order to more fully understand the scope of the problem and the service functions required. In contrast, a real use case concretely describes a service process in terms of its real current design and committed to specific technologies. Ideally, real use cases are created during the service design phase of a service development cycle, since they are a design artifact.

After the identification of use cases and concurrently with their specification, a use case diagram is created. It illustrates a set of use cases for a telematic service, the actors, and the relation between the actors and use cases. The purpose of this diagram is to present a kind of context diagram by which one can quickly understand the external actors of a telematic service and the key ways in which they use it.

3.2. Service analysis

The aim of this phase is to determine the functionality needed for satisfying the service requirements that were identified in the previous 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].

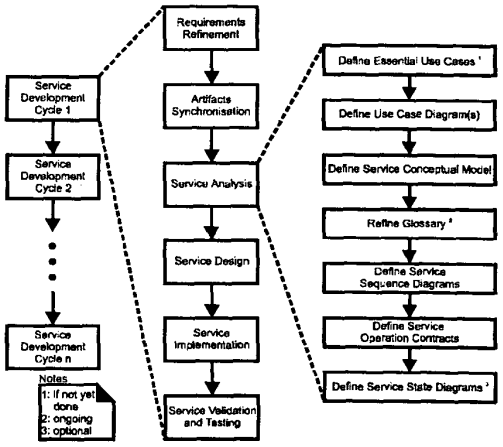

Figure 4. Service analysis phase activities

The activities that take place in this phase can be seen in Figure 4. As with the requirements capture and analysis phase artifacts, 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 during the service analysis phase can be seen in Figure 5.

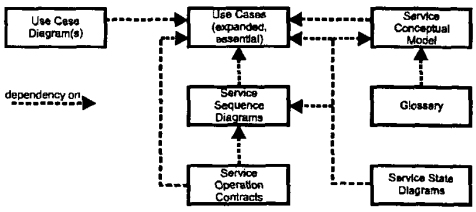

Figure 5. Service analysis phase artifact dependencies

The service analysis phase is the first phase of the service creation process where the service is decomposed into 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. Therefore, it describes what the service is in terms of interesting and meaningful (to the service developer) entities that constitute it and couplings between them. 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.

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 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 sessions and service sessions, and to the classification of access and service sessions. The most important of them is the Service Session Graph (SSG), which offers a generic framework to describe information in service sessions and is used to model and control the state of a service session.

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 are service sequence diagrams.

Use cases suggest how actors interact with the telematic service under examination. During this interaction an actor generates events to the telematic service, requesting some operation in response. It is desirable to isolate and illustrate the operations that an actor requests of a telematic service (service operations), because they are an important part of understanding service behaviour. A service sequence diagram, which is a UML sequence diagram, shows, for a particular scenario of a use case, the events that external actors generate and their order. All telematic services are treated as a black box. The emphasis of the diagram is events that cross the service boundary from actors to telematic services. 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.

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. UML contains support for defining service contracts by allowing the definition of pre- and post-conditions of service operations.

3.3. Service design

During this phase the service developer defines the behaviour of the service IOs that were identified in the service analysis phase and structures the telematic service in terms of interacting service computational objects (service components), which are distributable, multiple interface service objects. Therefore, the output of this phase is the dynamic view of the internal structure of the telematic service.

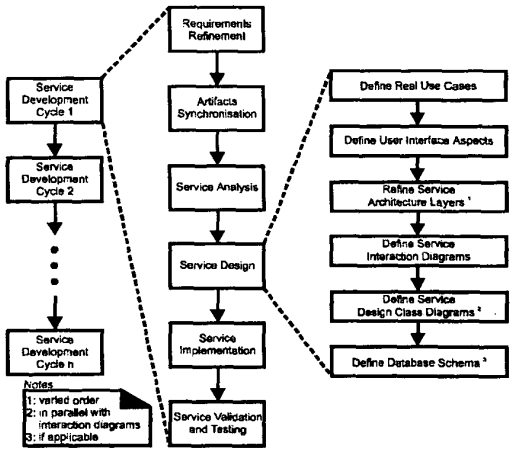

Figure 6. Service design phase activities

The activities of the service design phase are depicted in Figure 6. As with the previous phases, the linear order that may be inferred from this figure is not strictly the case, as some artifacts may be made in parallel (e.g. the service interaction diagrams and the service design class diagram). The dependencies between the artifacts produced during the service design phase can be seen in Figure 7. This figure also shows the way that the service design phase artifacts depend on some of the service analysis phase artifacts.

As a first step in this phase, the service IOs are considered as potential candidates for service Computational Objects (COs). In many cases, service IOs are

mapped to one corresponding service CO encapsulating the information defined by the service IO and providing an operational interface to access that information. However, the mapping between service IOs and service COs is not necessarily one to one. Furthermore, the existence of a relationship between service IOs, either provides a good rationale for encapsulating them together in the same service CO or indicates the need for a binding between interfaces of their corresponding service COs [4][5].

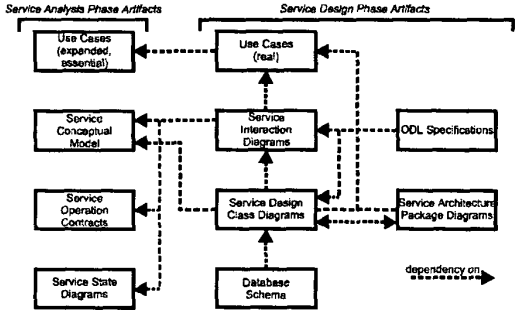

Figure 7. Service design phase artifact dependencies

This mapping process is significantly simplified by adopting the use of the generic (access session, service session, and communication session related) COs proposed by the TINA-C service architecture (in terms of their identified functionality and not in terms of specific interfaces - feature sets) and by considering the computational views of a number of scenarios (regarding business administrative domains in user-provider roles and peer-to-peer access roles, and in compound service sessions) deduced by the computational modelling guidelines of TINA-C, which are useful (for improving structure and comprehension) throughout the service design phase.

After identifying the service COs a (separate) service interaction diagram is created for each service operation under development in the current service development cycle. Service interaction diagrams illustrate how service objects communicate in order to fulfill the service requirements. The service designer may collect / extract information about what tasks the service interaction diagrams should fulfill by essential or real uses cases, and by the post-conditions of the service operation contracts. However, it is essential to recognise that the previously (in the service analysis phase) defined post-conditions are merely an initial best guess or estimate of what must be achieved and they may not be accurate.

Another important artifact created during service design is the service design class diagram, which illustrates the specifications for the software classes of a telematic service using a strict and very informative notation. More specifically, from the service interaction diagrams the service designer identifies the software classes (service classes) that participate in the software realisation of the telematic service under examination, together with their methods, and from the service conceptual model the service designer adds detail to the service class definitions. It has to be noted that in practice, service design class diagrams and service interaction diagrams are usually created in parallel. Furthermore, in contrast with a service conceptual model, a service design class diagram shows definitions of software entities (service components), rather than real-world concepts.

3.4. Service implementation

During this phase an implementation of the telematic service (service code) is generated from the service specifications and the deployability of the overall implementation on a TINA-C compliant DPE is examined (DPE targeting). It is assumed that at the beginning of this phase a specific (object-oriented) programming language and a specific distributed object platform are chosen.

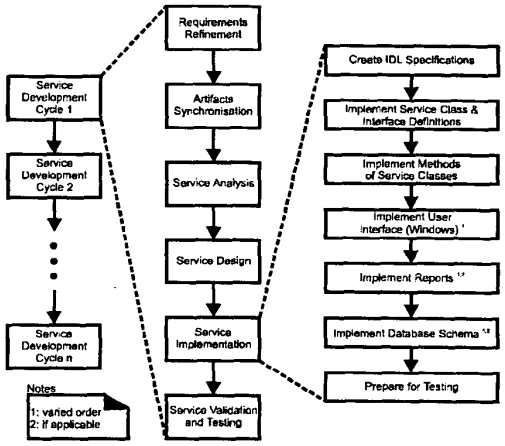

Figure 8. Service implementation phase activities

The activities of the service implementation phase can be seen in Figure 8. The dependencies between the artifacts produced during this phase can be seen in Figure 9. This figure also shows the way that the service implementation phase artifacts depend on some of the service design phase artifacts.

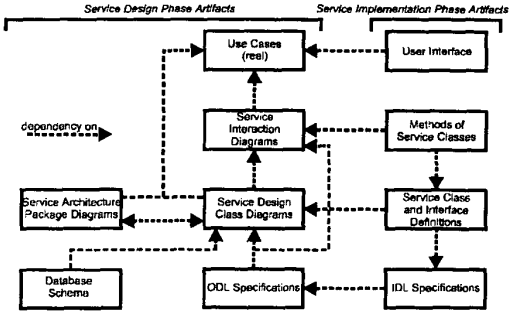

Figure 9. Service implementation phase artifact dependencies

The engineering representation of a service CO (using an object-oriented programming language like C++ or Java) is called an engineering Computational Object (eCO). The mapping between service COs and their eCOs is one to one: no eCO represents a composition of service COs nor is a service CO represented by more than one eCOs. The interfaces of an eCO represent the interfaces of its corresponding service CO [7].

With the completion of the service design phase there is sufficient detail to generate code for the service objects and construct the appropriate eCOs. For this purpose the service design phase artifacts provide a significant degree of the necessary information and the translation process is relatively straightforward, especially if service classes are implemented (and tested) from the least coupled to the most coupled. More specifically, as a service interaction diagram shows the messages that are sent in response to a method invocation, the sequence of these messages translates to a series of statements in the method definition. Furthermore, from the service design class diagram, a mapping to the basic attribute definitions and method signatures is almost evident.

3.5. Service validation and testing

Validation takes place in this phase by comparing the developed service software against the service specifications produced at the service design phase. This activity can be subdivided into the following two subactivities:

- *Conformance testing:* It involves checking the implementation for conformance to architectural rules and standards used in the service design.
- *System testing:* It comprises the testing of service software in a (possible) operational environment.

With this phase a service development cycle ends and another one (depending on the exact nature of the specific service requirements) is ready to start. A significant strength of the iterative and incremental service development process adopted by the proposed methodology is that the results of a prior service development cycle can feed into the beginning of the next service development cycle. Thus, subsequent service analysis and service design results are continually being refined and informed from prior service implementation work. For example, when the code in cycle N deviates from the service design of cycle N, the final service design based on the implementation can be input into the service analysis and service design models of cycle N+1. For this reason, as can be seen from Figure 1, an early activity within a service development cycle is to synchronise the created artifacts.

4. Conclusions and Future Work

In this paper, the key features that should characterise an advanced service creation methodology were presented and examined. This methodology enforces the service developer to take into account all the necessary features for the successful design and realisation of a service by exploiting the session-oriented, multi-party, multi-domain nature of the TINA-C service architecture. It provides a step by step approach from problem definition to the realisation of telematics services. For this purpose, the service is studied and described (in a number of service development cycles) at hierarchically related abstraction levels, in the sense that at each level the results achieved at previous levels of abstraction are preserved and refined.

The proposed methodology is being applied to the development of a MultiMedia Conferencing Service for Education and Training (MMCS-ET) using Microsoft's COM/DCOM as a DPE [2][6]. More specifically, a variety of scenarios are 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). The results obtained so far provide confidence that this methodology is useful for the description and development of (complex) new telecommunications services. However, further experience on the application of the methodology is necessary, together with the establishment of a collection of evaluation criteria and metrics for more comprehensive verification procedures.

Under these conditions the proposed service creation methodology is expected to enrich TINA-C and enforce it to pave the way towards an open integrated broadband telematic infrastructure populated by a virtually limitless variety of telematic services.

References

[1]D.X. Adamopoulos, G. Haramis, and C.A. Papandreou, "Rapid Prototyping of New Telecommunications Services: A Procedural Approach", *Computer Communications*, Vol. 21, 1998, pp. 211-219.

[2]D.X. Adamopoulos, G. Pavlou, and C.A. Papandreou, "Supporting Advanced Multimedia Telecommunications Services Using DCOM", *Proceedings of IS&N 2000*, LNCS, Vol. 1774, Springer-Verlag, Berlin, 2000, pp. 89-104.

[3]G. Booch, J. Rumbaugh, and I. Jacobson, "*Unified Modelling Language User Guide*", ACM Press, 1998.

[4]M. Declan, "Adopting Object Oriented Analysis for Telecommunications Systems Development", *Proceedings of IS&N '97*, LNCS, Vol. 1238, Springer-Verlag, Berlin, 1997, pp. 117-125.

[5]P. Demestichas, et al., "Issues in Service Creation for Open Distributed Processing Environments", *Proceedings of ICC '99*, June 1999.

[6]C.A. Papandreou, and D.X. Adamopoulos, "Design of an Interactive Teletraining System", *BT Engineering*, Vol. 17, pp. 175-181, 1998.

[7]TINA-C, "*Definition of Service Architecture*", 1997.

[8]TINA-C, "*TINA Business Model & Reference Points*", 1997.