# STRUCTURING DESIGN ACTIVITIES IN OPEN PROGRAMMABLE NETWORKS

Dionisis X. Adamopoulos
*Department of Technology Education & Digital Systems, University of Piraeus, Greece*

Keywords: Service design, new telecommunications services, service engineering, service creation, UML.

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 design 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 design are highlighted and a validation attempt is briefly outlined.

## 1 INTRODUCTION

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. 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) (Berndt, 2003), (TINA-C, 2003), is proposed.

## 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 (Adamopoulos, 2003).

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.

In the following paragraphs the service design phase of the proposed methodology is examined focusing on its essential characteristics and artifacts.



X. Adamopoulos D. (2008).
STRUCTURING DESIGN ACTIVITIES IN OPEN PROGRAMMABLE NETWORKS.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 260-263
DOI: 10.5220/0001522402600263

# 3 THE SERVICE DESIGN PHASE

During this phase the service developer defines the behaviour of the service concepts (service Information Objects, IOs) that were identified in the service analysis phase and structures the telematic service in terms of interacting service computational objects (service components or service objects), which are distributable, multiple interface service objects. They are the units of encapsulation and programming. While service IOs mainly explain how a service is defined, service Computational Objects (COs) reveal what actions have to be performed in order to execute the service. Therefore, the output of this phase is (mainly) the dynamic view of the internal structure of the telematic service.

Initially important characteristics of the user interface of the service are defined by examining the related prototype (produced during service analysis) and taking into account the feedback from the users of the service. The adherence to specific GUI standards and user interface design principles is also decided in this activity. The application of the Model-View separation principle, according to which the service logic should not be bound to a particular user interface, is proposed (Constantine, 2005)(Larman, 2006).

After identifying the service COs, by taking into account the service conceptual model(s) and the TINA-C service architecture, 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 fulfil the service requirements. More specifically, initially the expanded use cases suggested the service events which were explicitly shown in service sequence diagrams, then an initial best guess at the effect of these service events was described in service operation contracts, and finally the identified service events represent messages that initiate service interaction diagrams, which illustrate how service objects interact via messages to fulfil the required tasks.

Therefore, service interaction diagrams reveal choices in assigning responsibilities to service objects. The responsibility assignment decisions are reflected in the messages that are sent to different service objects. Responsibilities are related to the obligations that a service object has in terms of its behaviour. In the service implementation phase, methods will be implemented to fulfil responsibilities or alternatively responsibilities will be implemented using methods, which either act alone or collaborate with the methods of other service objects.

UML defines two kinds of interaction diagrams, either of which can be used to express similar or even identical message interactions; namely collaboration diagrams, which illustrate object interactions in a graph or network format, and sequence diagrams, which illustrate interactions in a kind of fence format (Evits, 2006). The use of collaboration diagrams for the expression of service interaction diagrams is preferred over the use of sequence diagrams, because collaboration diagrams are characterised by expressiveness, an ability to convey more contextual information (such as the kind of visibility between service objects), and a relative spatial economy.

Nevertheless, either notation can express similar constructs. What is really important is that service interaction diagrams is one of the most significant artifacts created during both service analysis and service design, because the skilful assignment of responsibilities to service objects and the design of collaborations between them are two of the most critical (for the satisfaction of the service requirements and thus for the successful realisation of a service) and unavoidable tasks (which also require the application of design skill) that have to be performed during service creation (Larman, 2006).

This activity of the service design phase consists mainly from the following steps:

*Step 1:* Identify the service COs.

During this step, the service IOs depicted in the service conceptual models (main and ancillary) that were created in the service analysis phase are considered as potential candidates for service 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 (Declan, 2000), (Demestichas, 2004). This mapping process is significantly simplified by adopting the use of the generic (access session, service session, and communication session related) service COs, proposed by the TINA-C service architecture

(TINA-C, 2003), in terms of their identified functionality and not in terms of specific interfaces/feature sets.

*Step 2:* Consider the generic TINA-C service scenarios and select the most appropriate.

After identifying the service COs and before proceeding to the construction of the service interaction diagrams, the computational views of a number of generic TINA-C service scenarios, deduced by the computational modelling guidelines of TINA-C (TINA-C, 2003), should be considered. These are useful for improving structure and general comprehension throughout the service design phase, and for offering to the service developer(s) a generic pattern of thought, compatible with fundamental TINA-C concepts, that he/she could use/consider when designing the service interaction diagrams.

*Step 3:* Form the service interaction diagrams.

A telematic service is composed of a set of service COs interacting with each other via messages with the objective to complete the required service operations. The service operation contracts present an initial best guess at responsibilities and post conditions for the service operations. Service interaction diagrams illustrate the proposed design solution (in terms of service COs) that satisfies theses responsibilities and post conditions, and therefore the corresponding service operations.

A service interaction diagram in the form of a UML collaboration diagram is created for each one of the service operations that were identified in the service analysis phase. The objective is to fulfil the responsibilities and the post-conditions of the corresponding service operation contracts, recognising however that their accuracy should be questioned.

As was explained in step 1 of this activity the service COs that participate in the service interaction diagrams are drawn from the service conceptual model(s). Therefore, the links between them are actually instances of the associations present in the service conceptual model(s), represent connection paths between service object instances, and indicate that some form of navigation between the instances is possible. More specifically, in order for a service object to send a message to another service object it must have visibility to it. Thus, it is important to ensure that the necessary (attribute, parameter, locally declared or global) visibility is present in order to support the required message interaction (Jacobson, 2006) (Larman, 2006).

Finally, all telematic services have a "Start Up" use case and some initial service operation related to the starting up of the telematic service. Therefore, there should also be a "Start Up" service interaction diagram, which is proposed to be created last. Although the "Start Up" service operation is the earliest one to execute, the development of its service interaction diagram should be delayed until after all other service operations have been considered. This ensures that significant information has been discovered concerning what initialisation activities are required to support the "Start-Up" service operation interaction diagram. The way that a telematic service starts and initialises is affected by related concepts/guidelines in the TINA-C service architecture (e.g. it is assumed that the IA must be present at the provider domain), and is dependent upon the DPE, the programming language, and the operating system that is being used.

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(s) the service designer adds detail to the service class definitions.

A service design class diagram typically includes/illustrates service classes, their attributes and methods, attribute type information, navigability, and associations and dependencies between service classes. 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.

In the service design phase, Specification and Description Language (SDL) can be used to describe the behaviour of a telematic service exploiting the finite state machine concept. Then, the SDL specification will serve also as a basis for validation, simulation and test case generation (Combes, 2005). In general, for making formal models of telematic services and being able to reason about these models, SDL is undoubtedly the notation of choice, as the tool support for SDL is perhaps the most advanced of all the formal notations existing today. However, adopting an SDL-based approach cannot guarantee that the developed services will be error free and the value of SDL for service creation purposes is questioned, as it may introduce unnecessary complexity in the service design phase.

Furthermore, the application of SDL can be difficult (or even problematic) in the case of relatively complex telematic services with many service objects interacting in non-trivial ways, due to the problem of state space explosion.

In the service design phase, service COs have a dominant role. Their interfaces are the result of the examination of the service IOs and the corresponding information models that they participate in, which reveal the way that service IOs are related to each other. This aggregation of interfaces into a service CO ensures the semantic understanding that operations at one interface may affect the behaviour of other interfaces because they may be linked by a common, underlying information model captured by the service CO. Therefore, such information models influence considerably the parameters and the semantics of the operations found on the interfaces of the service COs.

In order to aid the service development process TINA-C, proposes and prescribes a set of generic interfaces for the generic TINA-C service COs. These interfaces correspond to the interactions that take place between business administrative domains, support a particular session role, and are defined by the appropriate reference point specifications. TINA-C assembles the proposed interfaces into feature sets (TINA-C, 2003).

# 4 CONCLUDING REMARKS

Real use cases are members of the service design use case model, and service interaction diagrams are members of the service object behaviour model, because they describe the behaviour of service COs, and service design class diagrams compose the service class model. Furthermore, for reasons of completeness, the service design model includes service state diagrams for service COs / classes as members of the service design state model. Such diagrams may be useful to summarise the results of a service design (at the end of the service design phase) or when the service code is to be produced with a code generator that will be driven by the state diagrams.

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 Microsofta's Visual C++ together with Microsoft's Distributed Component Object Model (DCOM) (Adamopoulos, 2002) (appropriately extended with a high-level API in order to support continuous media interactions) as a distributed object-oriented environment.

# REFERENCES


Adamopoulos, D.X., Pavlou, G., Papandreou, C.A., 2003. Advanced Service Creation Using Distributed Object Technology. In *IEEE Communications Magazine*, Vol. 40, No. 3, pp. 146-154.

Adamopoulos, D.X., Pavlou, G., Papandreou, C.A., 2002. Continuous Media Support in the Distributed Component Object Model. In *Computer Communications*, Vol. 25, No. 2, 2002, pp. 169-182.

Berndt, H., Hamada, T., Graubmann, P., 2003. TINA: Its Achievements and its Future Directions. In *IEEE Communications Surveys & Tutorials*, Vol. 3, No. 1.

Combes, P., Renard, B., 2005. Service Validation. In *Computer Networks*, Vol. 31, No. 17, pp. 1817-34.

Constantine, L.L., Lockwood, L.A.D., 2005. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley.

Declan, M., 2000. Adopting Object Oriented Analysis for Telecommunications Systems Development. In *Proceedings of IS&N '00*, LNCS, Vol. 1238, Springer-Verlag, pp. 117-125.

Demestichas, P.P., et al, 2004. Issues in Service Creation for Open Distributed Processing Environments. In *Proceedings of ICC '04*, Vol. 1, pp. 273-279.

Evits, P., 2006. *A UML Pattern Language*. Macmillan Technology Series.

Jacobson, I., Booch, J., Rumbaugh, J., 2005. *Unified Software Development Process*. Addison-Wesley.

Larman, C., 2006. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall.

TINA-C, 2003. *Service Architecture*. Version 5.0.