A Framework for the Development of Service Management Systems for the Open Service Market

by

David Edward Lewis

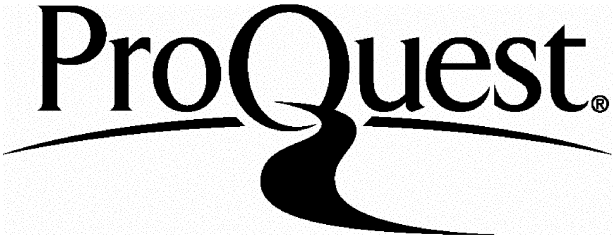
Department of Computer Science

University College London

A Thesis for the Degree of Doctor of Philosophy

Supervisor: Graham Knight

Date: 1st March 2000



Abstract

An open market in telecommunication services requires standard management interfaces to ensure interoperability between the operational support systems of the increasing number of market players generated by liberalisation. In addition, increased competition between these players heightens the needs for the rapid, cost-effective development of telecommunication management systems. Service Management bridges the gap between the network management activities of a service provider and its customer care activities and overall business aims. Systems implementing service management functions, therefore, must be well integrated with other service and management systems in the same and different administrative domains. However, service management suffers from a lack of standards for suitable architectures and technologies, as well as being exposed to the varied and volatile requirement imposed by a highly competitive and fluid services market. A common Development Framework for Service Management Systems (SMS) therefore needs to accommodate a range of architectures and technologies. The main benefit of a common development framework, therefore, may be in the development of a common development methodology. For a common methodology to be usefully and widely applied it must support the requirements of the main stakeholders in service management system development, namely SMS developers, the vendors of commercial off-the-shelf software these systems may use and the developers of the interface standards to which they may conform.

This thesis evaluates a variety of techniques for their suitability for a common development methodology that address the interactions and commonalities between development processes within the different stakeholder types. The techniques evaluated include those suggested in existing TMN recommendations, those using Open Distributed Process principles and those from existing general-purpose object-oriented analysis and design methodologies. The evaluation is based both on a review of the previous application of these techniques to management system development and to their application in a series of SMS development case studies in

which the author was involved. Some of the latter case studies introduced a level of empirical evaluation absent from previous studies

The result of the evaluations is the validation of specific techniques, namely, Use Cases, the Unified Modelling Language, the Analysis Modelling and Component Facades concepts developed originally by Ivar Jacobsen and the integration of Business Process and Role Modelling techniques during requirements analysis. A common methodology is described in terms of UML meta-models for integrated Business Requirements Modelling and a specialisation of the facade construct, termed a Projection. These are presented with examples and descriptions of how they may apply to the development processes of the different stakeholders. In addition, compatible, but loosely prescribed architectural guidelines are provided.

Acknowledgements

This thesis would not have been possible without the help, support and inspiration of a great many people. Firstly I would like to thank Graham Knight, my supervisor, for his patient guidance and good advice throughout the course of this project. I would also like to thank my second supervisor, Wolfgang Emmerich, and Jon Crowcroft who acted as my second supervisor for the first years of this work and who was a constant source of useful advice on the process of performing research. I would also like to thank George Pavlou, David Griffin and Saleem Bhatti, who together with Graham formed the core of a highly knowledgeable, world class telecommunications management research group at UCL, which provided an enormously stimulating environment for my research.

This work was conducted while I was working in a number of European Union sponsored projects so I also would like to acknowledge the many international colleagues with whom I have worked. In particular I would like to thanks Lennart Bjerring, Willie Donnelly, Jane Hall, Michel Louis, Jurgen Schneider, Lars Bo Sorensen and Michael Tschichholz for their insight and guidance in examining the development of TMN systems in PREPARE and subsequent projects. I would also like to thank Vincent Wade for his close collaboration on working on development methodologies in the Prospect and FlowThru projects. Many thanks also needs to go to the development team members who worked on the PREPARE, Prospect and FlowThru projects and who acted as guinea pigs for various development techniques addressed by this thesis. In particular, I must thank the developers at UCL for their assistance and feedback, namely: James Cowan, Alina DaCruz, Anne Hutton, Chris Malbon, Alistair McEwan, Rong Shi and Thanassis Tiropanis.

Finally I thank my family and friends for their help and encouragement in this arduous and time-consuming endeavour. In particular, I thank my Mum and Dad for their unconditional love and support, Fernando Urquidi for inspiring me to do a PhD in the first place and Liz Dancer who's praise, affection and advice helped make the completion of this work possible.

Table of Contents

Table of Figures References

Table of Table References

1. Introduction

The liberalisation of telecommunications markets around the world in the late 1980s and the 1990s has led to an explosion in the number of market players. Telecommunication markets, which were previously characterised by unchallenged national monopolies, have now become globalised and intensely competitive. As a result, the major new pressures that have come to bear on the providers of telecommunications services are, as identified in [adams]:

- The need to compete by *lowering prices*: which exerts a general downward pressure on costs and spur to automate processes within service providers.
- The need to compete on *quality*: where the speed and efficiency with which a service provider responds to a customer's orders, queries and problems can be a key competitive differentiator.
- The need to compete using *new services*: which is accelerated by the convergence of telecommunications industry with other sectors such as data communications, cable TV and publishing and the lowering of the cost of entry for new players, such as mobile operators and Internet Service Providers, due to technical innovations [beamount].

Standardisation is encouraged so that open interfaces to services, products and components may help prevent customers being tied to service providers and providers being tied to equipment and software vendors. In addition open interfaces aid providers in collaborating to provide services and thus help to prevent incumbents from excluding new entrants. Open interfaces are also developed through the collaboration of industry players to encourage the adoption of a new technology [leakey]. For a competitive market in telecommunication services underpinned by the use of interface standards the term *Open Service Market* is used here.

Returning to the pressures faced by service providers, one of the major operating costs they experience is that of *managing* their networks. Networks need to be managed to ensure that they service the end points required by customers, that the throughput and utilisation of the network is optimised and that equipment faults are dealt with as and when they occur. Such activities are categorised as *Network Management*, and are a major cost in the operation of a network. Software systems, termed *Network Management Systems* (NMS), are employed to help automate this task in order to reduce the cost of network management. This is an area that has long been addressed by open standards, e.g. Open Systems Interconnection - Systems Management (OSI-SM) and Simple Network Management Protocol (SNMP) [stallings][hegering]. Many conformant products are available and have been deployed by service providers

In a competitive environment, to allow a provider to make a profit while both controlling costs and providing superior customer service, network management must be conducted in close collaboration with *Service Management*. Service management is the term given to the management of the communications facilities offered by a network to provide a commercial service to customers. Service management involves tasks such as: mapping customer orders efficiently onto network provisioning and configuration activities, monitoring the delivered quality of service against the levels of service agreed with individual customers and mapping customer complaints onto network faults or performance problems. In addition to supporting such customer-driven needs, the network usage information made available by NMS is required both to generate charging information with which to bill the customer, and to monitor the utilisation of the network in order to plan network growth [varley]. Service management, therefore, represents a further major cost for service providers and there is a pressing need for them to provide effective Information Technology (IT) support for these activities if service providers are to be competitive. The software systems that implement service management activities are termed *Service Management Systems* (SMS).

SMS development has not benefited from standardisation to the same extent as NMS development. This can be attributed to several factors:

- While network management standards have emerged from the need for providers to have open interfaces to manage network elements, the needs of service management, i.e. service management interoperability between providers and the open procurement of SMS software, have only recently emerged due to liberalisation.
- Service management features such as billing and customer care, are often key competitive differentiators, so providers are often reluctant to collaborate in standardising them.
- The wide adoption of network technologies such has ATM and TCP/IP has provided a common base for open interface modelling for NMS. Services, which change rapidly in response to competition, do not offer a similarly stable model of what SMS must manage.

Though service management requires some level of distribution in its implementation, it does not have the extreme requirements for distribution of network management. So though standard network management technologies such as OSI-SM and SNMP could be applied to service management, they do not serve key requirements. As a result, many of the general purpose distributed processing platforms emerging in the IT sector present alternative, cost-effective solutions for service management. Amongst those that have been suggested as the basis for service management solutions are; technologies from the World Wide Web [maston], the Object Management Group's Common Object Request Broker Architecture (OMG CORBA) [chen96], the Open Software Foundation's Distributed Computing Environment (OSF DCE) [gaspoz] and intelligent mobile agents [corely]. It is therefore not currently feasible to advocate a standard service management platform technology that is likely to be widely accepted. This is explored further in Section 2.3.

The SMS development community is therefore faced with a potential vicious circle. The rapidly changing nature of the market imposes wide-ranging and volatile requirements on SMS that retards the development of the standards that could simplify problems of interoperability and encourage off-the-shelf solutions. Meanwhile the lack of a common framework for the development of SMS leads to the development of ad hoc solutions that in turn exacerbates the conditions that retard the development of common approaches. It could be argued that this situation is no different to that in any industry sector that relies heavily on IT. However, the highly inter-connected nature of the telecommunications industry means that obstacles to inter-operability are real barriers to competition and collaboration and thus to healthy growth. As a result the industry has a long tradition of overcoming these obstacles through the definition of open interfaces. However, the close relationship of service management to fast changing business and software environments (as opposed to the more stable network environment), indicates that a different approach to achieving open systems may be required in this area of telecommunications.

Shared system development problems, such as this, are often addressed by the effected community establishing a *Common Development Framework.* In telecommunications management, it is widely recognised that the lack of a coherent development framework has often resulted in poor integration of operational support systems, high levels of duplication due to stove-pipe solutions for different services and networks and thus high cost of ownership and operation [furley][adams]. A development framework that is open also allows for the integration of solutions from external vendors.

A development framework is typically an organised set of architectural and methodological guidelines for system development, coupled with an extensible set of reusable parts that perform functions useful in the problem domain (see Figure 1-1). *Architectural guidelines* are based on knowledge of the problem domain and aim to help practitioners to divide the problem domain up into manageable parts in a consistent way. Architectural guidelines address both the logical architecture and the

technological architecture applicable to the problem domain. The *logical architecture* consists of a structured division of the logical functional areas addressed by the development framework, with layering being a common structural technique. The *technological architecture* establishes the common underlying IT functionality required to implement solutions. The more common functionality that can be identified in a domain and bundled into a supporting IT platform, the easier it is for developers to address the specific problem at hand. The *set of reusable parts* enables the developers to draw on the knowledge and experience gained from previous solutions in the problem domain. The *methodological guidelines* provide the processes and notations needed by developers to build systems that solve their particular problems in a way that is consistent with the framework's architectural and technological guidelines and which make best use of and possibly contribute to the set of reusable parts. *The methodological guidelines of such a framework are main focus of this thesis.*

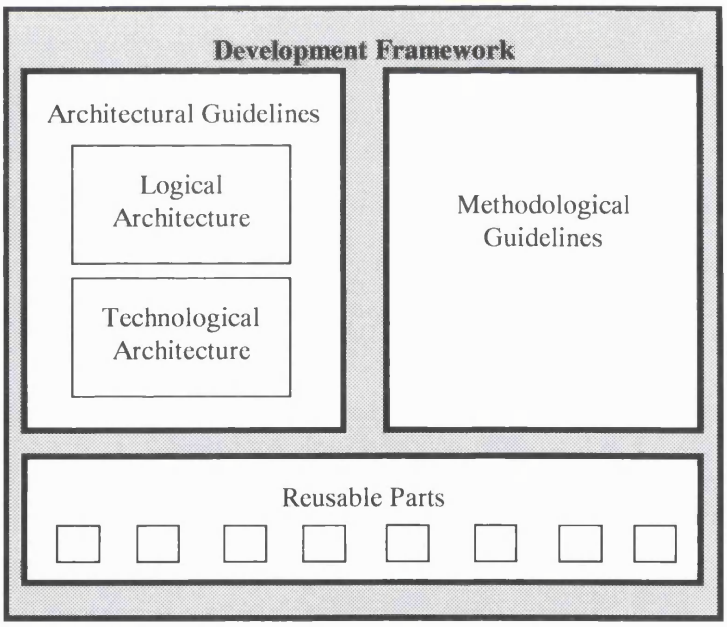


Figure 1-1: Generic Structure of Development Framework

Given the above considerations the thesis can be stated as follows:

The broad ranging and rapidly changing environment of the Open Service Market, in which SMS must be developed means that a common, durable, well-populated architecture for service management is currently unattainable. A development framework suitable for SMS must, therefore, have a loose, extensible logical architecture encompassing a range of different technologies.

Within such a framework, a common, well-understood approach to modelling system requirements and design will enable developers to deal with the complexity and heterogeneity of SMS within cost and time constraints. To be effective, such a common modelling approach must take into account the methodological requirements of SMS developers, of the developers of open interfaces to which SMS conform and of the developers of commercial software used in SMS. The approaches currently taken to the development and standardisation of NMS are insufficient to support the development of and standardisation for SMS in the Open Service Market. However, existing techniques from the Open Distributed Processing and Software Engineering communities can be successfully applied to these requirements.

1.1 Approach and Contribution

The thesis proposes that a common methodological guideline is the key part of an open development framework for SMS that must necessarily present only loose architectural guidelines. The development and testing of such an open development framework was conducted based on the analysis of a novel *business stakeholder model*, presented in Section 2.1 of the problem definition in Chapter 2. This model addresses the individual needs and required relationships between the three main stakeholder types involved in the development of SMS. These are the developers of SMS, the vendors of the commercial off-the-shelf (COTS) software components that may be used within SMS and the developers of open interface standards to which SMS and COTS may conform. This model forms the core of the approach taken to defining a suitable open development framework for the thesis. Chapter 2 completes the definition of the problem presented in defining an open SMS development

framework by analysing the current state of *standards* addressing service management (Section 2.2) and the range of *technologies* that may be reasonably applied to service management (Section 2.3). The range of standards analysed consists of international standards for Telecommunication Management Networks (TMN) and Intelligent Networks, the work of the TeleManagement Forum and that of the Telecommunications Information Networking Architecture Consortium (TINA-C). This provides evidence that no common logical architecture for service management is likely to emerge in the near term. The range of applicable technologies presents a similar conclusion for the prospects of a common service management technology. Based on this analysis of the current context for SMS development, a series of goals for an open SMS development framework is synthesised (Section 2.4).

A comprehensive *survey of existing methodological techniques* that have been suggested and applied to the development of SMS, of management interface standards and of management systems in general is then provided, in Chapter 3. This survey covers popular *general purpose software engineering techniques* (Section 3.1), techniques associates with *Open Distributed Processing* (ODP) standards (Section 3.2) and techniques developed specifically for *telecommunication management development* (Section 3.3). An analysis of this survey (Section 3.4) reveals that there is a low level of objective evaluation of these techniques and no empirical evaluation. However, commonalities in the structuring of these development approaches are used in Section 3.5 to define a model of the development processes that typically are performed by each of the SMS development stakeholder types. This model also identifies the relationships between development processes within the same stakeholder type and between processes in different stakeholder types. This yields another novel aspect of the approach taken in this thesis, in that it is both the *commonalities* and *relationships* between development processes in different stakeholders that are highlighted as offering the widest benefit from commonality in the techniques applied to them.

Next, in Chapter 4, the *core experimental work* of the thesis is presented, where different development techniques are evaluated to assess them against the overall goals for an open SMS development framework and specific requirement of the SMS developement stakeholder process model. As is observed in Chapter 3, however, evaluation of SMS development techniques suffers from the general problem experienced in validating software-engineering techniques. Firstly, measures of desirable software features such as reliability, maintainability and flexibility are difficult to define [fenton]. Also it is recognised that, in addition to the methodology followed and the tools used, the effectiveness of a software development effort is highly dependent on the people involved and the characteristics of their personal interactions. These human factors are difficult to control, and at best can be randomised, leading to limits to the replicatability of and therefore the validity of many software experiments. Software development is also a very expensive activity due to the value of software developer's time. Therefore, fully controlled experiments tend to be small in scale and often don't reflect a real commercial software development environment, e.g. experiments performed on groups of students. Alternatively, experiments can be carried out within commercial software development projects. Here, budget and time constraints often take priority over experimental considerations, limiting the control of variables and the collection of data. This thesis presents a particular problem since open service management is a relatively new area and not yet a common basis for software development. In addition, as the effectiveness of SMS development is crucially linked to the effectiveness of interactions between SMS Developers, Software Vendors and Standards Developers, an experiment within a single software development organisation alone would not address the full problem domain. For a PhD thesis study the option of performing a replicated, controlled experiment with a significant number of subjects is ruled out due to the cost involved. The approach taken, therefore, was to conduct a series of case studies where SMS development activities were observed and the experiences of SMS developers garnered. The case studies were conducted as SMS development work in collaborative research projects. Here,

groups of developers from universities, research institutes and industry, worked together in realising networks of integrated management systems, including SMS, in order to publicly demonstrate novel aspects of telecommunications management. Though these development projects do not fully reflect real commercial development, their collaborative nature makes them a closer approximation to the multi-stakeholder open service market environment. In addition, these projects were more amenable to experiments attempting different development methodologies.

The approach to evaluating the methodologies used in each case study varied from anecdotal observation of developers' experiences, through group discussion of such experiences to structured questionnaires. In all but the first case study, the author had team leader responsibilities for the modelling, implementation and integration of SMS, and was therefore in a good position to garner on-going, informal feedback of the techniques used, as well as to organised the more formal evaluations. The main aim of the evaluations was to assess the practicability and relative usefulness of different development approaches applied to SMS development.

The Case Studies provide the following experimental information:

- Case Study 1 used OSI-SM technology for the integration of network and service management in a multi-provider environment. It provided anecdotal evidence of the problems involved in defining a multi-interface system using the notations and processes defined for TMN by the ITU-T. It also provided some initial evidence of the power of *scenario* and *business role* definitions for SMS development.

- Case Study 2 addressed the same problem domain, but provided anecdotal evidence of the benefits of *modelling the responsibilities* between roles to supplement scenario descriptions. It also provides anecdotal evidence of the benefits of using computational modelling, as practised in TINA, to functionally decompose systems.

- Case Study 3 involved a more complex enterprise model than the previous case studies, with the SMS were implemented using the Object Management Group's

Common Object Request Broker Architecture (OMG CORBA). It provided evidence of the usefulness of *ODP viewpoints*, as applied in TINA, through anecdotal feedback, a questionnaire and a group discussion. Weaknesses were found with the ODP approach however, in particular in maintaining consistency between the information and computational viewpoints.

- Case Study 4 extended further the multi-domain system developed in Case Study 3, but using *Use Cases* and the *Unified Modelling Language* (UML). The experiences of developers of the usefulness of these techniques for multi-domain system, single-domain system and component development was captured via a questionnaire.
- Case Study 5 built on the results from Case Study 4 and refined the application of UML to support more fully the reuse of components in systems that implemented management business information flows. Jacobsen's *analysis modelling technique* [jacobsen92] and his *facade modelling construct* [jacobsen97] were used to model components in a more reusable manner, while UML *activity diagrams* were used to help model business process interactions. The case study was evaluated though a questionnaire.

Chapter 5 presents the results of the thesis in terms of *methodological guidelines* for an open development framework. The evaluation of the individual case studies and the state of the art analysis of suitable methodologies was synthesised into a set of *methodological recommendations*. These commend the use of *Use Cases*, the *UML*, Jacobsen's *analysis modelling* abstractions and his *facade modelling* construct. The recommendations are accompanied by specific modelling guidelines in the form of extensions to the UML meta-model, which represent the core contribution of the thesis. These meta-model extensions address:

- A simple but precise definition of the components of a *Use Case Description.*
- The definition of a *Business Requirements Model*, which combines use case modelling, role and responsibility modelling, business process modelling and multi-domain system modelling.

- The definition of a modelling construct called a *Projection*. This is similar to Jacobsen's facade construct, but with the use of Jacobsen's analysis modelling abstraction defined more precisely.

The application of these modelling constructs to the development process interactions between the SMS development stakeholder types is then presented.

Chapter 6 presents areas were this work could be extended. It addresses: further extending and validating the meta-model; aligning the models with emerging software component architectures as they might apply to SMS; providing CASE tool support for the integration of the modelling constructs with other open, telecommunications modelling notations and applying the modelling concepts to the future standardisation of service management capabilities.

Finally Chapter 7 concludes the thesis by assessing the results against the thesis statement and the goals set out in Chapter 2.

As the case studies were collaborative exercises the author was not solely responsible for defining the methodologies used. In Case Studies 1 and 2 the methodologies applied were defined collaboratively by a core of the development team, which included the author. For Case Studies 2 and 3 the methodologies used were defined jointly and equally by the author and Vincent Wade of Trinity College Dublin. The application of the business process to business role mapping and the facade modelling construct in Case Study 5 was specified by the author alone. The subjective evaluation of development techniques in the case studies is based both on the author's own experiences of applying the techniques and on the informal feedback gained from other developers individually and through group discussion. The empirical evaluations conducted for Case Studies 4 and 5 were entirely the author's work, as were the problem analysis of Chapter 2, the state of the art analysis of Chapter 3, the methodological recommendations and guidelines of Chapter 5 and the assessment of further work in Chapter 6. A reference to publications where the author has already presented material in this work is given alongside that material.

To summarise, this thesis performs some empirical analysis of the applicability of various software development techniques to the specific needs of stakeholders involved in SMS development. The results of this analysis are used to define some UML modelling constructs that, though generally applicable to open, multi-domain, component based system development, are focussed on the development process interactions identified between the SMS development stakeholder types.

2. Problem Definition

This chapter first defines a model of the organisational types that have a stake in the development of SMS. It then reviews the current state of the art of service management in terms of the available standards and applicable technologies. A set of goals is then defined establishing an open SMS development framework.

2.1 Stakeholders in SMS Development

To fully understand the requirements for an open SMS development framework, a good understanding is needed of the organisational types that have a stake in SMS development. Many of the requirements are driven by the relationships the involved organisational types may have with each other. In order to fully understand these relationships and the resulting requirements a suitable organisational stakeholder model must be established. Several models have been proposed by various fora and researchers, which attempt to provide a basis for analysing the various business entities involved in telecommunications management and the relationships between them. The approach typically taken does not identify different existing organisational types since the structure of the sector is changing too rapidly for the businesses involved to be easily and durably classified. Instead, business roles are identified in the hope that most businesses can then be classified as playing one or more roles. Examples of role-based organisational models are the TINA Business Model [mulder], the business model used in the EURESCOM project P.610 [nesbitt], the enterprise model used in the RACE project MOBILISE [keil] and the TMF's Business Reference Model [nmf-gb910]. These models reflect the different business concerns and technology choices of the group generating the model. For instance TINA is heavily based on ODP principles, part of which is the concept of a trader that directs consumers of services to providers based on a description of the service required and other non-functional parameters such as cost. Hence TINA identifies a Broker business role, not present in other models, that performs an ODP trader-like activity. Also the P.610 and MOBILISE business models differentiate between a

public network operator role and an access network provider role, reflecting real divisions in the market place, while TINA opts for a more abstract view with a single connectivity provider. The roles identified by these are summarised in Table 2-1.

Role	TINA	P.610	MOBILISE	TMF
Customer/Subscriber	X	X	X	X
End User		X	X	
Service Provider	X	X	X	X
Connectivity Provide	X	X	X	
Access Provider		X	X	
Third Party Provider	X	X	X	X
Service Broker	X			
Application Vendors				X
Equipment/System Suppliers				

Table 2-1: Comparison of Business Model Roles

For assessing the requirements for a SMS development framework, a suitable business model does not need to detail operational roles in more detailed than needed to identify their impact on the development process. It is necessary; however, to identify and highlight the relationships of stakeholder who have a direct impact on the SMS development process. Though all of these models are part of frameworks that intend to support the on-going development of telecommunications systems, only the TMF model explicitly identifies a system supplier role. It also identifies a third party application vendor to acknowledge that third party software is an increasingly important way of reducing development costs.

The following business roles are suggested as ones with a significant stake in SMS development. These roles are inspired by those identified in the TMF model, but are

tailored to aid the identification of requirements for an open SMS development framework:

- *Service Provider:* A role that provides commercial telecommunication services to a Service Customer.
- *Third Party Service Provider:* A role that the Service Provider must collaborate with in order to provide the required service to the customer, e.g. underlying services, content provision, peer services which extend coverage.
- *Service Customer:* A role that consumes and pays for the service provided by a Service Provider.
- *SMS Developer:* A role that undertakes the development of the SMS required by a Service Provider to manage its services.
- *Software Vendor:* A role that develops software to sell on the open market.
- *Standards Developer:* a role that develops standards related to service management functions and relevant computing platforms.

The relationships between these roles have been limited to those that relate to the development of SMS, and thus influence the requirements on the development framework.

The *SMS Developer* role develops bespoke SMS for the Service Provider and the Service Provider may select different organisations playing the SMS developer role for different SMS development projects, including possibly the maintenance of previous SMS from other developers. Typically these roles have been played by the same organisation, but the model separates these roles in the anticipation that SMS development will be increasingly out-sourced. The SMS Developer is motivated to reduce SMS development costs and delays and SMS maintenance costs in order to secure continued SMS development contracts from the Service Provider.

The *Service Provider* role is principally concerned with controlling the costs and improving the quality of the service management it delivers, as well as managing new services. It also agrees cost and time constraints for SMS development with the

SMS Developer role. The Service Provider is the main source of functional requirements for the SMSs that are developed by the SMS Developer. These requirements may include adherence to specific open standards as part of an overall business strategy. However these requirements are also shaped by the relationships the Service Provider has with other market players. For instance, a Service Provider may not actually own and operate a network but may be part of a service provision chain, as discussed in [davidson94][noam]. For the purpose of analysing the requirements for a development framework it is sufficient to ignore the complexities of possible value chains and simply consider the roles of Service Customer and the Third Party Service Provider relative to the Service Provider. Business relationships with these stakeholders may also include agreements to adhere to open interface standards for operational interaction at the service management level. The current lack of inter-domain service management standards and the concomitant scarcity of off the shelf products to support such interactions mean that the collaborative agreement and implementation of interoperable interfaces between SMS Developers must be addressed by the model. Though the development of SMS for the Service Customer and the Third Party Service Provider are outside the core focus of the model, this relationship introduces the requirement for the SMS Developer to interact with another SMS Developer contracted to provide an interoperable SMS to the Third Party Service Provider.

Software reuse is widely seen as a key technique in meeting the challenges of developing software within cost and time constraints [jacobsen97][karlsson]. One of the most promising techniques is the reuse of Commercial-Off-The-Shelf (COTS) software developed by a third party, modelled here as the *Software Vendor* role. With COTS software reuse, the costs of developing and maintaining the software component falls on the Software Vendor, who recoups it by selling the component to as many customers as possible, the customer in this model being the SMS Developer. In addition, the SMS Developer can also practise software reuse internally by reusing solutions to frequently occurring problems, or by implementing standard interfaces popular with SMS customers.

The *Standards Developer* will develop standards mostly in response to the petition from the other roles. The Service Customer may petition for open service management interfaces that enable it to move between Service Providers easily, thus encouraging competition between them. Service Providers and Third Party Service Providers may petition for service management interface standards to ease the implementation of multiple third party relationships or to meet regulatory interoperability requirements. The SMS Developer is motivated to support these standards, as they will promote the reuse of software that implements them between different SMS development projects. The Software Vendor will also have an interest in management interface standards since they reduce the risk of investing in the development of COTS software that conform to such standards. Both the SMS Developer and the Software Vendor will also have an interest in using standards that promote software portability and integration, e.g. CORBA.

This stakeholder model therefore emphasises the need for open standards to underpin a competitive market both in the delivery of service management and the provision of SMS software. The model is novel in identifying the Standard Developer role as one that should be addressed by a development framework for open service management. This recognises: that there is no dominant forum for service management solutions; that work from many different fora are relevant to SMS development and that an open development framework must therefore support multiple standards developers. The interactions between the roles in the stakeholder model are shown in Figure 2-1.

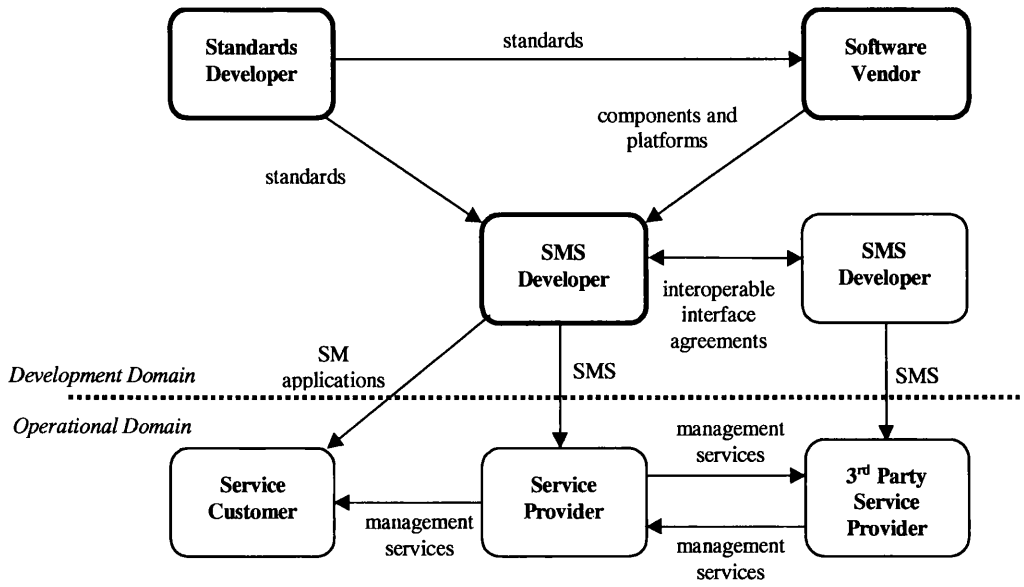


Figure 2-1: Summary of SMS development stakeholder roles and their relationships

The author believes that an approach to the design of a development framework driven by the needs of software developers and standards developers rather than being driven by architectural and technological consideration is a novel one and will result in a more flexible and robust development framework.

2.2 Open Service Management

This section reviews existing open development frameworks that address service management. It describes how each framework addresses service management and also the structure of its functional architecture and scope of its population of reusable parts.

2.2.1 Service Management in TMN

To date, several standards generating bodies have addressed the area of service management. One of the first standards to draw the distinction between service management and network management was the Telecommunications Management Network (TMN) series of recommendation from the ITU-T [m3000]. The TMN architecture is specified in recommendation M.3010 [m3010] and defines a Logical Layer Architecture in which conceptual layers address different concerns within a

provider's management network. The following layers are identified (listed from the bottom up):

- A Network Element Layer (NEL) containing the network resources to be managed.
- An Element Management Layer (EML) that is concerned with the management of individual network elements.
- A Network Management Layer (NML) that is concerned with managing individual networks.
- A Service Management Layer (SML) that is concerned with the management of customer services.
- A Business Management Layer (BML) that is concerned with the management of the entire enterprise.

Each layer is intended to provide the layer above it with the functions required to perform its functions. Therefore, we can define the service management layer as being involved in using network management layer functions both to support the delivery of services to customers and to provide the functions needed by the business management layer. M.3010 also states that the service management layer is responsible for maintaining statistical information, interactions with other service providers and interactions between services.

M.3010 specifies a functional architecture that identifies types of functional blocks and the types of reference points that exist between them. The taxonomy of functional blocks makes the distinction between: a Network Element Function (NEF) managing individual network elements; a Mediation Function (MF) that mediates between different TMN interfaces; a Q-Adapter Function to non-TMN compliant network element managers; a Work-Station Function (WSF) presenting information to human operators; and a general Operations System Function (OSF) that monitors, co-ordinates and controls telecommunications functions. The functional architecture identifies reference points that define the functions that may

be exchanged between functional blocks. Reference points therefore provide the basis for defining interfaces between physical implementations of the functional blocks. The functional architecture also distinguishes between the types of reference points connecting functional blocks within a single TMN (q reference points) and those connecting OSFs in different TMNs (x reference points). TMN management functions are categorised, for the purpose of standardisation, into the areas of Fault management, Configuration management, Accounting management, Performance management and Security management. This categorisation is commonly referred to by the acronym FCAPS. A functional decomposition of the TMN problem area can therefore be represented as a five by five grid consisting of FCAPS in one axis and the TMN layers in another [des403-1]. This can also be further decomposed on a third axis according to whether inter or intra domain issues are being addressed, i.e. whether the resulting reference points are x or q type (see Figure 2-2).

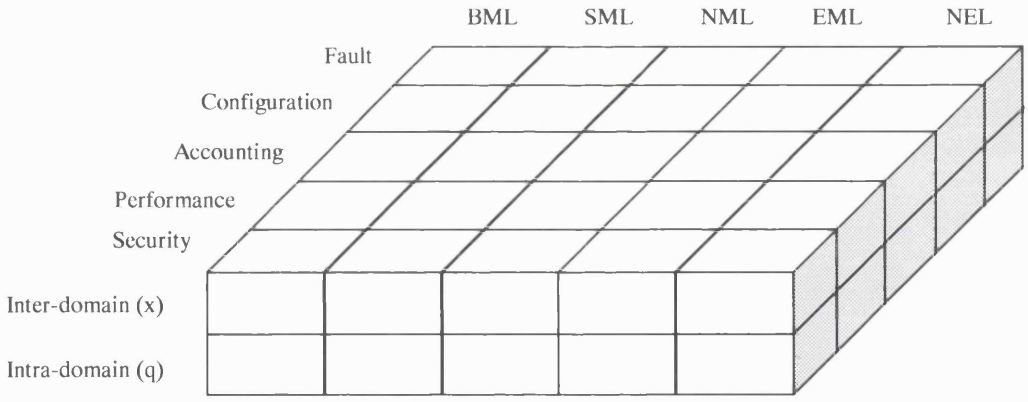


Figure 2-2: TMN separation of functional concerns

TMN interfaces are defined according to the methodology described in recommendation M.3020 [m3020-95], which results in the definition of management services, management functions and management information models. This methodology is discussed further in the next chapter. The TMN management services defined to date in [m3200] are mostly network-related, with the exceptions of Customer Administration and Tariff, Charging and Accounting Administration.

The more detailed TMN management functions defined in [m3400] also mostly address the network and network element management layers, though the following functions sets can be placed in the service management layer:

- Under Performance management there are function sets for subscriber service quality criteria, customer service performance summary and customer traffic performance summary.
- Under Fault management there are function sets addressing service outage reporting, arrangement of repairs with the customer and the area of trouble administration?
- Under Configuration management there are function sets addressing demand forecasting and the area of service planning and negotiation, which involves customer interactions, request for service functions and service status administration.
- Under Accounting management there are several function sets related to the areas of tariffing, pricing, collections and finance.
- Under Security management there are a few function sets addressing customer security alarm, customer profiling, customer usage pattern analysis, administration of customer revocation list, protected storage of customer data, customer audit trail and customer security alarm management as well as many functions that apply equally to customers and internal operations staff.

Some of management functions have been refined into information models, though mostly for function in the EML and NML, e.g. the generic network management in [m3100]. Some generic systems management functions exist in the form of OSI-SM System Management Functions [x700].

Despite its network management focus, examples exist of TMN being applied successfully to the architectural structuring of SMS for X.400 services [caluwe] and broadband Virtual Private Network (VPN) services [bjerring94b][griffin95].

2.2.2 Service Management for Intelligent Networks

One area where service management has been investigated in relation to standardised services, is that of the management of Intelligent Networks (IN), as standardised by the ITU-T in the Q.1200 series of recommendations [q1200]. Initial research into this area suggested that functional entities defined in IN standards for performing management functions could be implemented as TMN OSFs [magedanz][pavlou95a]. This work raised the issue of how OSFs access via CMIP, which is not optimised for real-time operations, could be integrated with signalling technologies that satisfy the real time requirements of IN service control. This revealed the lack of a clear functional demarcation between service management and service control within the standards for both (i.e. TMN and IN respectively). Problems were also identified with determining standardised solutions should handle non-functional requirements. Much of the current research in this area advocates the migration of both IN and TMN to a middleware based approach such as the one proposed in the Telecommunication Information Networking Architecture (TINA) [chapman], which is discussed in more detail below. IN service management analysis has already influenced TINA, with a set of intra-domain management requirements for IN Capability Set 2 [etsi-na608] being reflected in the design of the TINA Subscription Management model.

2.2.3 Service Management in the TeleManagement Forum

One body that has analysed service management requirements more directly is the Network Management Forum (NMF), which has recently been re-branded as the TeleManagement Forum (TMF). This not-for-profit, industrial forum aims to build on existing TMN standards with business agreements and procurement guidelines that directly address the industry's near-term needs. This is in recognition of the fact that the ITU-T's TMN standards address mainly network and network element management from a bottom-up perspective, and are therefore difficult to apply directly to business problems related to service management. In addition, problems have been found in applying TMN standards in their existing form to specific

management problems. These experiences led to the definition by the NMF in 1995 of the Service Management Business Process Model [nmf-gb901]. This was a result of a survey of several service providers in an attempt to identify key business processes involved in the provision and operation of services. The aim was to establish a provider-independent model that would help engage providers in the process of developing business agreements by reducing the potential for revealing sensitive proprietary process solutions. This model was supplemented in 1998 with a more detailed analysis of the business processes involved in network management [nmf-gb908]. These business models were combined and released as the TMF's Telecoms Operation Map (TOM) [nmf-gb910], which contains the business process model shown in Figure 2-3.

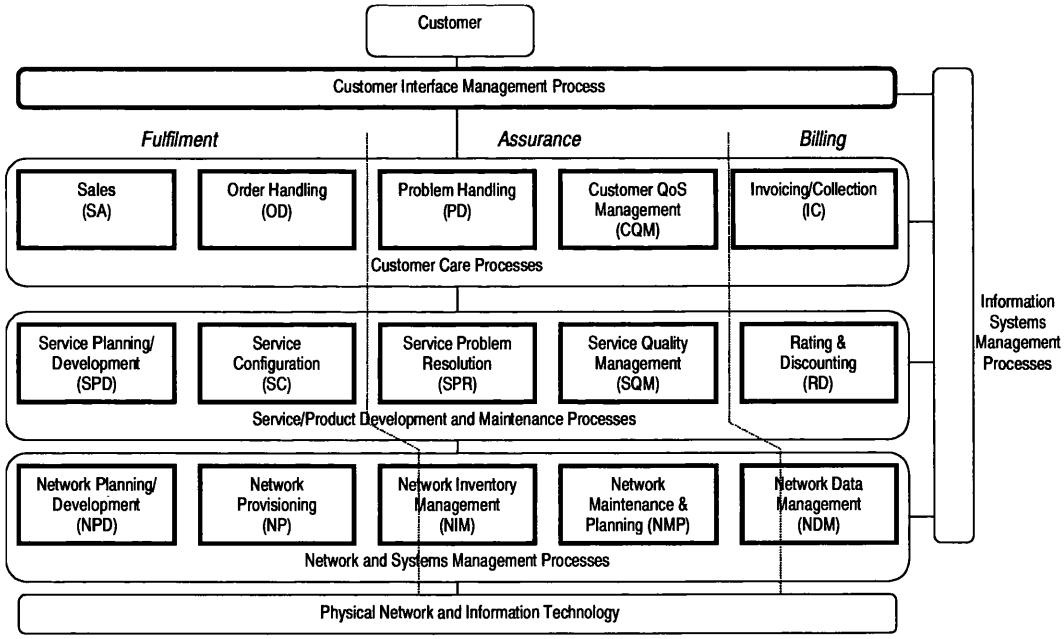


Figure 2-3: TMF's Telecoms Operation Map

In this model, the Customer Interface Management process, the Customer Care processes and the Service/Product Development and Maintenance processes can be regarded as those performing service management. The vertical divisions shown in Figure 2-3 represent a broad functional partitioning of the processes into those

involved in service delivery (i.e. fulfilment) those involved in maintaining the services (i.e. assurance) and those involved with billing for the service. These three vertical process groups correspond to the FCAPS categories of configuration management, performance and fault management combined and accounting management respectively. The TOM describes each process in more detail identifying activities performed within each process and the input and output triggers that pass between processes, with those between processes potentially in different service provider domains being highlighted.

This is the most comprehensive high-level functional architecture for service management in the public domain, certainly amongst the ones claiming to be open. As it is directly based on surveys of current service providers, the TMF regards this as a living document that will be updated every few years to reflect changing approaches to service provisioning and operation. Based on this model, the TMF is performing ongoing work in defining open management interfaces, both between service and network management processes, and amongst service management processes. Of the latter the following interface agreements are completed:

- Trouble Ticket - Electronic bonding: allowing service providers to exchange trouble information
- Trouble Ticket - Customer/Provider interface: enables customer SMS to receive trouble reports automatically from a service provider's SMS.
- Performance Reporting: provides common model for exchanging Service Level Agreements (SLA) information between customers and service providers.
- Order Exchange: provides a generic mechanism for exchanging service ordering information between service providers.

2.2.4 Service Management in TINA

Another body that has performed in-depth studies of service management is the Telecommunication Information Network Architecture Consortium (TINA-C). This industrial consortium has aimed to develop a comprehensive architecture for

telecommunications control and management based on the Open Distributed Processing principles defined by ITU in [x901]. TINA-C has generated a detailed development framework for the integrated control and management of multimedia and information services, principally over broadband networks [chapman]. It has drawn heavily on the recommendations from TMN, the IN standards and ATM signalling and management standards from the ATM Forum and ITU-T. TINA is split into four constituent architectures addressing network, service, computing and management concerns. The Network Architecture [tina-nra] and the Service Architecture [berndt95a] provide the main sources of detailed functional specifications within TINA. The Computing Architecture specifies a Distributed Processing Environment (DPE), which is the distributed, object-oriented software platform over which all TINA functions are provided [graubmann]. The Management Architecture [delafuaente94][delafuente95] defines the principles under which telecommunications management is performed in TINA, though the actual functionality is defined in the Network and Service Architecture specifications. The TMN logical layers are used as a basis for TINA's own logical layering of management functionality. The Management Architecture has a Service Management layer corresponding to the TMN SML. Below that it has a Resource Management layer which encompasses the TMN NML but also includes the management of the connectivity requirements for multimedia communication session. Finally the Element Management layer encompasses the TMN EML. TINA does not address the TMN BML. Figure 2-4 shows the scope of Telecommunications Management in TINA, together with the scope of Computing Management, which is applied to the applications performing service and management functions as well as the supporting DPE and computing and networking infrastructure.

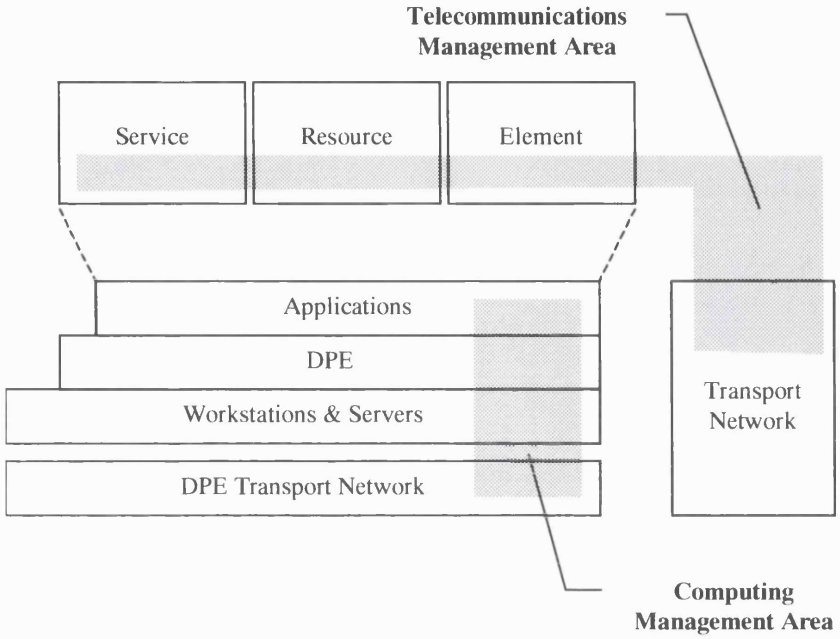


Figure 2-4: Management Areas in TINA

Management concerns within each layer are addressed within the FCAPS categories, however TINA extends their scope beyond those traditionally assumed for management. Specifically, in the Resource Management layer, configuration management is split into resource configuration, similar to the TMN interpretation of network configuration management, and connection management, which maps onto control aspects of the network which are typically addressed through signalling standards in the ITU-T and ATM Forum. This is due to TINA's assumption that the same DPE infrastructure is used for both management and connection control and hence classifies the latter as management also. A similarly close integration between management and what elsewhere is regarded as service control is present in the TINA Service Management layer as defined in the Service Architecture. This is advantageous in some respects since it enables the same concepts and mechanisms for interacting with users to be applied to both service delivery and service management. The distinction between control and management functions is

maintained in the definition of TINA reference points where they are allocated to core service segments and ancillary segments respectively [mulder].

Detailed service management functions are defined within TINA in the Service Architecture. This specifies detailed models for Subscription Management and Accounting Management. Subscription Management covers service life-cycle management, i.e. the introduction, suspension, and withdrawal of services composed of different service features, as well as customer life-cycle management, i.e. the subscription of customers to services and the authorisation of users to use these services. Accounting management addresses the collection of service and resource usage information, its assembly into charges and its distribution as bills. Subscription and accounting management are both tightly integrated with the TINA session concepts, around which the rest of the Service Architecture is structured. This principally is integration between subscription management and the access session, which mediates which services a user can invoke, and between accounting management and the service session, which controls the distributed execution of the service. Fault, Performance and Security Management are not yet directly addressed in the Service Architecture.

2.2.5 Current Status of Open Service Management

From this review of current open standards, it can be seen that there is no single accepted definition of service management. Though TMN addresses service management directly, it provides no clear definition of the boundaries of the SML and little in the way of functional content for this layer. The TMF Telecoms Operation Map provides a more complete view of the processes that are involved in the service management layer, though this model is not intended to be definitive, but an evolving contextual guide to their on-going efforts to generate open interface agreements prioritised by TMF members. Other attempts at defining service management functionality have been performed in support of specific open services such as IN and X.400. TINA presents a more generic open service definition based on the concept of sessions. While TINA services are connection-oriented and based

on the use of a DPE, research has shown that the session concepts can be adapted for use in environments with different network and computing technologies, e.g. the Internet [lewis98a][dezen98] and IN [herzog]. This implies TINA service management features may be widely applicable, though this is still a research issue. The relative immaturity of open service management standards, however, is indicated by the lack of evidence of these standards being used in industry. British Telecom for example, while accepting TMN in principle, has evolved its own operational support system architecture [furley].

In attempting to define service management based on open standards the following statements can be made:

- Service management is concerned with the activities within a service provider organisation that manage the delivery of telecommunication services to the customer.
- Service management makes use of network and network element management functions.
- Service management provides functions needed for the business management of service provider enterprises.
- Service management deals with customer sales queries, customer service orders, customer problems with service failure or performance and with charging the customer for the service.
- Service management is concerned with the deployment and withdrawal of services.
- Service management does not encompass the management of hardware or software components involved in the delivery of services. These are handled by network and systems management functions.
- Service management does not encompass management of the computing platform on which services or management services are provided.

Despite these common definitions, the fact that there is no widely accepted functional architecture that can be prescribed for an open SMS development framework means that the other portions of the framework must therefore support the ongoing development of multiple functional architectures and their growing populations of reusable functional units.

2.3 Technologies Applicable to Service Management

Many different technologies are currently being put forward as candidates for service management. The first management specific technologies were two rival manager-agent protocols, which were initially designed for managing network elements. Manager-agent protocols enact management on communications resources through their representation on an agent as a collection of Managed Objects (MOs).

One of the manager-agent protocols, SNMP [case], emerged from the Internet community. This protocol dominates the management of network elements in the data-networking sector but has not yet been applied to service management. To date, there has been very little interest in general from the Internet community in service management. There is, however, a growing recognition that issues of customer service profiles and billing [arkko] are key to the development of the Internet once IP-based services provide quality of service. Nevertheless, there is still little evidence of SNMP being applied to service management, with new protocols being developed in support of service management requirements instead [yavatkar].

The other manager-agent protocol is the Common Management Information Protocol (CMIP) [x711] underlying OSI System Management framework X700 series [x700]. This has been widely accepted by the telecommunications community, and was adopted for implementing the physical architecture in TMN, notionally including systems in the Service Management Layer. When applied to inter-domain management, which is typically enacted in the SML, the need was identified for CMIP to be integrated with X.500 directory in order to allow transparent navigation between MOs on separate agents [stathopoulos][bjerring94a]. The TMF has developed some service management related interface agreements using CMIP, and

several research projects have attempted to implement service management OSFs using CMIP platforms [hall96][griffin96][galis]. There is, however, little evidence that CMIP has been used for industrial service management applications to date. Research experiences in developing CMIP OSs reveal that the difficulties experienced were closely related to the CMIP Application Programming Interface (API) made available to the developer in the platform used. These APIs ranged from low level ones such as XMP/XOM used in Hewlett-Packards OpenView system, to the high level RMIB C++ [pavlou94] and Tcl/Tk APIs used in the OSIMIS platform [pavlou95b]. The differing nature of these APIs also precluded the reuse of code across platforms, though recently the TMF has produced an open C++ CMIP API [chatt].

Interoperation between managers and agents implemented using the different protocols is possible using gateways for converting between SNMP and CMIP [mccarthy][dassow]. However when accessing a CMIP agent from an SNMP manager via such a gateway, some of the protocol features of the more functionally rich CMIP, e.g. scoping and filtering, are lost.

Increasingly, the OMG's Common Object Request Broker Architecture [corba][chen97a] is being advocated for telecommunications management [stringer]. As with manager-agent protocols, CORBA provides client-server interoperability between remote systems through the use of standardised protocols, but without the built-in support for event notification and multiple object access available with CMIP. In addition, CORBA supports the standardisation of APIs for performing remote procedure calls from clients to server objects. This is done through standardising the mapping between various programming languages and the Interface Definition Language (IDL) used to describe CORBA server interfaces. Compilers can generate client stub and server skeleton code in a number of languages, e.g. C, C++, Java, Cobol, and Ada.

The OMG's Object Management Architecture (OMA), which encompasses CORBA [pope], therefore represents an open functional framework that could be applied to

SMS. The functional structuring within the OMA is not strongly prescribed, with the primary separation being into areas of increasing speciality. The separations are into categories of: generally useful CORBA Services [oma-cos] (e.g. the naming, event and persistence services); CORBA Facilities that are useful across application domains and Domain Facilities that provide domain specific solutions, e.g. the output of the OMG's Telecommunications Domain task force. Typically CORBA Facilities will make use of CORBA Services, while Domain Facilities will use both CORBA Services and Facilities.

The level of advocacy for the adoption of CORBA for service management varies. TINA-C has adopted the position that its DPE can be implemented using CORBA, and therefore that CORBA may be used for the control and management of both networks and services. There is a broad consensus, summarised in the TMF's Technology Integration Map (TIM) document [nmf-gb909], that the investment in network element management, and to a lesser extent, network management using CMIP and SNMP will result in these technologies persisting in these roles. This is reinforced by the features they provide for efficiently interacting with large numbers of managed objects distributed over large numbers of network elements, which are not available in CORBA. However there are strong arguments for CORBA to be adopted in the service management layer where CMIP does not yet have a strong foothold and so there is little or no CMIP based legacy to support. These arguments include: easier integration with legacy business systems, more likelihood of applicable CORBA applications emerging from the wider IT community, greater availability of CORBA development expertise and a wider range of platforms at lower prices. The growing popularity of CORBA has prompted the investigation of CORBA to CMIP and CORBA to SNMP gateways [deri], with standardised solutions now being available from the Joint Inter-Domain Management (JIDM) taskforce formed by the TMF and X/Open [soukouti]. An alternative approach to exploiting the TMN legacy that has been attempted for network management is to use the existing CMIP-oriented specifications to structure CORBA solutions [griffin97][potonniee].

The OSF's Distributed Computing Environment was an earlier distributed computing platform that was considered for use in telecommunication management, as described for example in [gazpos]. The OSF also built on DCE in developing the Distributed Management Environment (DME) which aimed to bring together network and system management in an object oriented manner. The use of DME was largely unsuccessful [marcus], and the use of DCE has now been largely superseded by CORBA.

Another recent technology that has received much attention for its applicability to management is the World Wide Web and downloadable application code in the form of Java applets. The low cost and sudden near ubiquity of WWW browsers makes them an attractive option for management applications. Several solutions have been found for using WWW browsers for browsing agents, e.g. [barillaud], while the Java to IDL binding enables interaction with CORBA-based management information.

To summarise, there seems little likelihood of a single distributed technology becoming dominant for the implementation SMS in the near future. However, technology gateways between candidate technologies seem to be feasible, and are being integrated into management platforms [rahkila][rasmussen], so this may not pose a major obstacle to SMS interoperability. It is clear, therefore, that the other portions of the development architecture must support the development of SMS on multiple different technological platforms. As distribution technologies are closely integrated with modelling techniques, e.g. CMIP with GDMO, SNMP with SMI and CORBA with IDL (see next chapter), the range of platforms that must be accommodated must be reflected in the range of interface specification techniques that must be supported in the development framework.

2.4 Synthesis of Requirements

Based on the analysis of the architectures and technologies that may apply to service management and the model of business roles that are involved in the development of SMS, this section lays out the goals for an effective open development framework for SMS.

The focal beneficiary of a development framework is the SMS Developer role as it is the support of this role that is the principle activity of the Standards Developers and the Software Vendors in this context. The SMS Developers are driven by the requirements imposed on them by their customers, i.e. the Service Providers who will use the SMS. The first goal can therefore be stated as:

Goal 1: *The Development Framework must support SMS Developers in developing SMS that satisfy the business needs of Service Providers, including its business interactions with Service Customers and Third Party Service Providers.*

To satisfy the first goal, the SMS Developer will gain the maximum benefit from a development framework if it addresses all its internal software development activities. Hence we can summarise the next goal as:

Goal 2: *The Development Framework must address all stages of SMS development, i.e.: requirements capture, system analysis, system design, systems testing, system deployment and system maintenance.*

The Service Provider operates in an open service market underpinned by open interface agreements that ensure the interoperability of its own SMS with those of its Customers and Third Party Service Providers. In addition, the SMS Developers wish to benefit from open IT solutions to enable the construction of system from solutions obtained from multiple different sources (i.e. Software Vendors). Therefore:

Goal 3: *The framework must support SMS Developers in the application of open standards from Standards Developers.*

The SMS Developers need to operate within cost and time constraints to ensure competitiveness and profitability. One approach to this is to build SMS products based increasingly from a portfolio of existing, internally developed software and to be able to exploit software bought of the shelf from Software Vendors. Hence the next two goals can be stated as:

Goal 4: *The Development Framework must support SMS Developers in the reuse of design solutions and software over different project.*

Goal 5: *The Development Framework must support SMS Developers in using commercial off the shelf software components developed by Software Vendors.*

For the Software Vendor, the risk of developing software for use by SMS Developers is reduced if that software product addresses known industry requirements as expressed in open industry agreements. Therefore a further goal is:

Goal 6: *The Development Framework must support Software Vendors in the application of open standards in developing its products.*

As we have seen in Section 2.2, no dominant functional architecture is emerging for service management, so the parallel development and co-existence of several functional architectures and constituent functional units must be supported. Therefore the development framework should be one that can be applied to the on-going development of standards within existing open development frameworks, including the continued use of standards not originally developed according to the common development framework. Hence:

Goal 7: *The Development Framework must support Standards Developers in the on-going development and evolution of open standards to be used by SMS Developers and Software Vendors.*

As we have seen in Section 2.3, a wide range of different technologies are applicable to service management, and are likely to co-exist thanks to interoperable gateways. Therefore a further goal is:

Goal 8: *The Development Framework must support the development of SMS that operate over heterogeneous computing platform technology and that will be robust to changes in computing platforms.*

Finally, a clear requirement for the widespread uptake of a development framework is that it is simple to understand and easy to use for practitioners. An important key to usability is the amenability of the framework to Computer Aided Software Engineering (CASE) tool support. A wide range of CASE tools have been applied to the development of management systems, which presents problems in terms of

integrating development processes and exchanging models [valiant]. A common development methodology must be deployable on as much of the installed base of CASE tools as possible, but should also facilitate the development of more suitable CASE tools if necessary. Therefore the final goal is:

Goal 9: *The notations and methodology of the development framework should be easy for those playing SMS development stakeholder roles to understand, and should be readily supported by CASE tools.*

2.5 Summary

To summarise, the lack of a common functional architecture described in Section 2.2 and the range of technological solutions applicable to service management described in Section 2.3 retard the definition of common architectural guidelines as part of the development framework. The effectiveness of any common development framework must therefore depend more on the common applicability of its methodological guidelines, rather than on that of its architectural guidelines. For this reason the thesis proposes that a common methodological approach to the whole SMS development problem will be more effective than attempting to synthesis yet another set of common architectural guidelines. The above goals will therefore be used in subsequent chapters to assess the suitability of existing methodologies for SMS development, to analyse the requirements for methodological guidelines in the development framework and to test, through case studies, different approaches that combine existing development techniques.

3. Analysis of Existing Frameworks and Requirements Synthesis

This chapter reviews the various existing software development methodologies that have been applied within the telecommunications domain and to the development of management systems in particular. A very wide range of software development methodologies exists, so this analysis is restricted to those that most closely match the requirements for an open SMS development framework given in the previous chapter. The methodologies addressed are ones that are part of de facto or de jure standards and/or ones that are closely related to the technologies applicable to service management as described in Section 2.3.

The review of methodologies is split into three sections. The first addresses methodologies that have emerged from the general software engineering community, in particular the ones that have been widely accepted or standardised or which have had a visible influence on standard management systems development methodologies. The second section addresses the methodologies that have come from the distributed systems community, in particular those related to Open Distributed Processing. The third section addresses methodologies that have been developed for telecommunication system development and for management system development in particular. Material presented in this chapter relating to M.3020, the TMF methodology and the TINA methodology is based on an existing survey by the author [lewis99c].

This chapter is concluded by a fourth section which attempt to synthesise the methodological techniques that have been analysed into a high level categorisation of development models and processes. This categorisation is mapped onto the needs of the SMS development stakeholders identified in Chapter 2.

3.1 General Software Engineering Methodologies

The last decade has seen the generation and promotion of a huge number of general software development methodologies, though to date no one methodology has emerged as dominant. However, two categories of methodology that have gained wide spread acceptance are Object Oriented (OO) Analysis and Design and Business Process Engineering. A further technique, Design Patterns, has also attracted much recent attention, and shows potential for aiding the communication of common solutions between developers. These different techniques are reviewed in the following sections together with their application to SMS development where relevant.

3.1.1 Object Oriented Analysis and Design

Object Oriented Analysis and Design (OOAD) methodologies use concepts originally developed for object oriented programming with languages such as C++ and Smalltalk. Most OOAD methodologies therefore share common concepts of; object instances containing state only accessible through a well-defined interface; object classes that defined the interfaces and the inheritance between object class definitions. The earlier Object Oriented (OO) methodologies focussed on design, where OO modelling provided benefits by localising design changes and thus minimising unexpected interactions. OO design also removed the problems related to having shared data areas and were considered well suited to distributed or parallel system implementations. Though OO designs do not have to be implemented in an OO programming language the mapping from design to implementation is much more straightforward if they are, with the generation of OO-language code from OO designs becoming a major benefit of OO design. OO analysis exploited OO concepts in the modelling of real world situations that were to be addressed by software systems. Typically problem domain artefacts were identified from requirements statements through techniques such as mapping nouns phrases to objects and verbs to methods on objects. As with the relationship between OO design and OO programming, OO analysis may be useful without integration with OO design

techniques, however integrated OOAD methodologies have emerged the most popular.

A wide range of OOAD methodologies has been generated, with Graham, in his 1994 book [graham] identifying over 28 distinct methodologies. One of the earliest and most influential OO methodologies was that developed by Grady Booch in 1991 and revised in 1994 [booch94]. Booch observes that all software development methods include: a notation for expressing the various models used; a process describing the ordering of development activities and tools to aid the developer follow process rules, reduce errors and maintain consistency. In Booch's methodology, the notation is structured as a logical model, expressed in class and object diagrams and a physical model expressed in module diagrams and diagrams showing the distribution of processes between processors. The dynamic aspects of the notation are provided by state transition diagrams for individual classes and interaction diagrams showing the flow of messages between classes. Booch advises that the development process should be both iterative and incremental, but acknowledges the need for waterfall-based project management to ensure progress is monitored and guided correctly. He therefore categorised the process into two parts. Firstly, developers cycle through a micro development process consisting of identifying classes and objects, identifying their semantics, identifying their relationships, specifying class and object interfaces and implementing them. Secondly, the project must follow a macro-process, which starts with establishing core requirements, developing a model of the desired behaviour (i.e. analysis), creating an architecture (i.e. design), evolving the implementation and subsequently managing post-delivery execution (i.e. maintenance). As pointed out in [graham], the Booch methodology focuses more on the design and implementation part of the overall development process and is weak in providing guidance for capturing and analysing user requirements, suggesting only that these be based on scenario descriptions. Though Booch's object oriented design modelling concepts were very influential, there is little evidence of the notation being used for telecommunication management related activities.

Another influential methodology that first appeared around the same time as Booch's was James Rumbaugh's Object Modelling Notation (OMT) [rumbaugh]. OMT places an emphasis on the object oriented modelling of concepts rather than just of a design solutions and thus can be categorised as an integrated OOAD approach. This methodology is structured around three models: an object model using object class diagrams, a dynamic model using nested state diagrams and a functional model using data flow diagrams that show how processes operate on data originating from individual objects or aggregate data stores. The process is split into three phases: systems analysis; system design, where the system is subdivided into more workable units and object design, where the objects and their relationships identified in the analysis are elaborated upon. Rumbaugh places a similar emphasis to Booch on using an iterative process, but places more emphasis on the analysis phase and on the dynamic and functional models. The rich expressiveness and clarity of the object modelling notation made it popular in many applications. It was applied to the modelling of management system by the TMF for a period (see later in this section) and was used by TINA-C for expressing models in the ODP information viewpoint (see Section 3.2 for further details). There are few examples of the full OMT methodology being applied to management. The application of OMT to VPN management service design in [chan-m] is unusual in that it uses the functional model but not the object model. Examples exist for IN development [dezen97][milsted]. In [milsted], it is observed that though the OMT dynamic model expresses the change in state of objects, it does not readily allow the representation of how relationships between objects vary over time. The solution proposed is to use a 'storyboard' that consists of an ordered sequence of object instance diagrams.

Another popular OOAD methodology is Object Oriented Software Engineering (OOSE) first described by Ivar Jacobsen et al in [jacobsen92]. This was first worked upon while Jacobsen was working for Ericsson, and is therefore well grounded in telecommunications software development. This methodology places more focus on the capture and analysis of requirements than Booch or Rumbaugh's. It introduces the concepts of use case modelling and analysis modelling. Use case modelling

involves describing textually the interactions of users with the system under analysis, thus providing a structured, semi-formal representation of the system's functional requirements. Use cases and their interactions with users, or actors as Jacobsen calls them, can be summarised in use case diagrams. Also use cases can be generalised, so that common patterns of user-system interaction can be captured and requirements therefore can be stated more concisely. Analysis modelling is a high-level form of object modelling where use case text is analysed to generate objects of three types. These object types are: interface objects dealing with the interactions between users and the system; entity objects modelling information in the system and control objects which operate on one or more entity objects and interact with actors via interface objects. Such high level modelling is claimed by Jacobsen to clarify the architectural issues of system development and thus contributed to a more robust design. Modelling with analysis object is therefore referred to as robustness modelling. These objects are represented as specialised icon in a robustness model diagram that showed static relationships between them such as inheritance and message channels. These objects and their relationships are then transformed into more general purpose design object diagrams. These are refined with the aid of sequence diagrams, which are driven by use case descriptions, and the consideration of the system's implementation environment, e.g. database and distribution requirements. Jacobsen claims that the power of this approach lays in use cases being the focus for modelling activities at all stages of development, i.e. for analysis modelling, design modelling, implementation and testing. He therefore claims that OOSE covers more of the development process than the Booch and Rumbaugh methodologies. Though use cases have been widely used in management related methodologies (see the TMF and P.610 approaches discussed later in this section and the application of use cases in Case Studies 3, 4 and 5 in Chapter 4), there has been little evidence of the analysis object types being widely applied.

Jacobsen further refined his methodology in 1997 in [jacobsen97], which applied use case and robustness modelling to software reuse. This approach uses a modelling construct called a facade which presents a view of a component for its reuse by using

relevant modelling elements from its use case, robustness analysis, design, implementation and testing models. Though there is little evidence of this construct being applied to management systems, it has been assessed in the SMS development context by the author in [lewis99b] and as part of Case Study 5 in Chapter 4.

The most significant potential impact these three methodologies have had on the SMS development problem has been through the evolution of the *Unified Modelling Language (UML)* [booch99][eriksson][fowler]. The development of UML was motivated by the realisation that though the processes used in the different methodologies varied, and often had to be adapted to specific problem domains, the notations contained many similar modelling concepts. UML was developed jointly by Booch, Rumbaugh and Jacobsen while working together at Rational Software Corporation. UML version1.1 has now been adopted as a standard notation for OOAD by the OMG [ad/97-08-03], which now also controls its evolution. The semantics of the language are expressed as a meta-model that defines the types of modelling elements that UML provides, the relationships that may exist between them and the constraints that are imposed on those relationships [ad/97-08-04]. UML also provides notational guidelines for the visualisation of models in a variety of diagram types [ad/97-08-05]. The diagram types are:

- Class diagrams visually similar to OMT.
- Object diagrams showing class instances.
- Use case diagrams based on OOSE.
- Interaction diagrams of two types, sequence diagrams showing interacting object instances as vertical bars and collaboration diagrams showing interactions between objects instances as depicted in object diagrams but connected by temporally enumerated message lines.
- State-chart diagrams showing the event driven state machine behaviour of a system.

- Activity diagrams, which are a specialisation of state-chart diagrams showing the flow of control between activities in a system.
- Component diagrams showing the organisation and dependencies of components and the interfaces that exist between them.
- Deployment diagrams that show the runtime configuration of processing nodes and the components that operate on them.

An additional strength of UML is its extensibility mechanisms, which allow the language to be extended in a controlled way. These mechanisms are: stereotypes, which allow new modelling elements to be derived from existing ones; tagged values, which allow the information handled by the modelling elements to be extended and constraints, which allow new semantic rules to be added or existing ones modified. The OMG supports a mechanism whereby process specific extensions for UML can be agreed. For example a proposal exists for supporting UML extension for the analysis object types found in Jacobsen's OOSE methodology [ad/97-08-06]. The definition of a suitable process to apply UML is intended by its authors to be a matter for individual problem domain groups to agree upon. Some general UML-based development processes are emerging [korthaus][allen][kivisto] as are initial experiences from industrial use of UML [hruby]. UML's standardised status has greatly accelerated its support within CASE tools and it is increasingly widely accepted as the standard notation for OOAD. It has therefore attracted attention from bodies such as TINA-C, TMF and EURESCOM as a suitable notation for defining management standards. However, the opportunity to agree common UML profiles or processes for telecommunication management has not yet been exploited.

3.1.2 Business Process Modelling

Though OOAD techniques have provided major benefits for software engineers, they are not always very accessible to the managers, customers and other players in a business who have to supply and agree the requirements of a system and understand its operation and its implications for their business activities. It is shown in [arlow]

for example that while non-technical managers, users and domain experts had a good comprehension of UML use cases, this fell off rapidly when dealing with class, interaction and state diagrams. *Business process modelling* is one increasingly popular approach to bridging the gap between describing the business needs of a company in a way that those involved in its operation can understand, and defining the requirements for IT systems that support these needs. Business process modelling is widely used to assist in general business process re-engineering, defining and controlling the flow of work within organisations, configuring standard software, developing bespoke software and performing activity-based costing. Notations such as Event-driven Process Chains (EPC) are used to depict: the relationship between business functions and events; the flow of control between functions and the flow of data between functions and the users or organisational units with which they interact. EPC also supports additional modelling features such as timing constraints on functions and probability of control flow options to enable the simulation of what-if scenarios when performing business process re-engineering and the determining of performance requirements for supporting IT systems. As pointed out in [allweyer], UML activity diagrams support some of the expressiveness of EPCs, but would need to be enhanced and better integrated with class and use case models to provide similar analytical power.

The main proponent of business process modelling for management systems is the TMF. Its Telecoms Operations Map is intended as a provider-independent process model for use in driving the agreement of common interfaces.

3.1.3 Design Patterns

The software engineering community has recently recognised problems with reuse techniques that rely solely on object-orientation. One response to these problems has been to adapt the concept of *Design Patterns* from the architectural and construction industry and apply it to software reuse as demonstrated by Erich Gamma [gamma]. Design patterns aim to capture the knowledge of experienced designers and document it in a manner that facilitates the communication of architectural

knowledge and known design traps to other developers. Typically, therefore, design patterns represent common, well-proven designs. There are several, slightly different forms suggested for documenting design pattern, however they all follow a common structure. At a minimum a design pattern states a problem and outlines a solution to it, with a context description that indicates the applicability of the solution. The latter part is key, since the aim is not to sell the pattern to the reader, but to provide the information needed by the reader to enable them to gauge whether the solution presented fits well to the problem with which they are faced. An important feature of a pattern is a suitable, and preferably brief, name. In this way, it is hoped that pattern-based terminology will evolve into a more powerful means for communicating between software developers. Great emphasis is placed on expressing patterns concisely and clearly. Ideally they should also contain an example of an application or coding of the pattern. Design patterns are typically collected together into Pattern Catalogues [coplien][vlissides], though more benefit can be gained for the user when a collection of related patterns, possibly addressing a specific application domain, are carefully cross-indexed, to show how the solutions can work together in different ways. Such an inter-related collection of patterns is referred to as a Pattern Language. Gamma's original pattern catalogue addressed how small groups of software objects solved common problems, however patterns have been written to address a wide range of problems up to and including patterns for structuring enterprises. Mowbray and Malveau [mowbray] suggest that patterns could actually be categorised into levels of architectural scale from those containing a few classes to those spanning several organisations.

Design patterns therefore show potential as a way of exchanging knowledge between developers, with examples of telecommunications solutions expressed as patterns becoming more common in the literature [aurrecoechea][meszaros]. However, it is less clear how patterns can integrate with the exchange of specific OOAD models. Though UML had a notation for expressing patterns, the author has found no solid examples of how this can be applied within a broader OOAD specification.

3.1.4 Relevance to SMS Development

The above general software engineering techniques vary in their relevance to SMS development. OOAD is clearly relevant to the development of SMS. However, as pointed out by Martin, based on experiences of applying OOAD the telecommunication systems development [martin], most OOAD approaches do not take into account the inclusion of models from sources that have not followed the same approach or used the same notation. This presents a problem within the context of SMS development stakeholder model presented in the previous chapter where models must be imported from both the software vendor and the standards developer. With respect to the latter, Martin recommends that the fora generating models for telecommunication interface standards should migrate to an OOAD approach so that the full benefits of these techniques could be exploited when integrating the resulting specifications into applications.

The TMF has explicitly recognised this problem in its internal modelling and design methodology [vincent]. This provides methodological guidance to those within the TMF responsible for developing open interface agreements. It describes the definition of the interface as an adjunct to the development of the systems that provide the solution to the problem for which the interface was required. This therefore involves consideration of the systems on both sides of the interface, rather than just the modelling of agent/server management services, management functions and managed objects as had been performed previously in management interface development (see Section 3.3 for more details). The TMF methodology borrows heavily from contemporary OOAD methodologies such as those described above. The process consists of the following stages:

- System Overview: This uses use case diagrams to define the context of the system. An interesting addition is to allow specific actor-system interaction flows to be identified at this stage as conforming to an existing standard. Also the interactions that are the focus of the interface definition effort are made distinct from those simply providing contextual support.

- Problem Statement Capture: This involves two techniques ideally performed with the co-operation of the potential standard user community:
  - The enumeration of requirement statements and their categorisation into ones relating to: structural information, dynamic information, abnormal conditions, expectations and non-functional requirements and system administration requirements.
  - Detailed use case descriptions focussing on the actor-system interactions under scrutiny, these include traces to requirement statements.
- Requirements Modelling: This involves refining the problem statements and use case descriptions to identify common pieces of information and to provide clearer descriptions of their usage and the functions that may be performed on them using the OMT models.
- Analysis Modelling: This involves transforming the requirements analysis into a system design by further refinement of the OMT models. It is recommended at this stage to use, if possible, existing design patterns or to use object types similar to Jacobsen's analysis objects in order to structure the design.
- Design modelling: This involves mapping the analysis model into its implementation environment which consists of a standardisation framework (the TMF's TOM being the obvious choice) and the system's technical environment such as operating systems and databases.

Unfortunately no information is available as to the effectiveness of this process as used by TMF development teams. However, the approach is being used and is rapidly evolving, with a revised version of the methodology, based on the use of UML as the notation, under development.

A further initiative by the TMF that builds on this methodology is that of *protocol neutral modelling*. This aims to define interface standards in terms of OMT class models and then automatically generate from these models interface definitions in the notation required, e.g. the OMG's IDL or the ITU-T's Guidelines for the

Definitions of Management Objects (GDMO) [x722]. To this end the TMF has defined mappings between OMT and GDMO and, as protocol neutral modelling has evolved to use UML, have done a similar mapping between UML and GDMO. Extensions to existing tools to support these mapping are also available. The aims of this approach are to make the interface modelling work robust to changes in distributed platform technologies and to allow them to be applied across a wider range of such technologies. As reported in [hall98] however, problems were experienced in supporting mapping from OMT to both IDL and GDMO due to differences in how they represent aggregation relationships, name bindings and relationships between objects, as well as the presence of specialised mechanisms such as scoping, filtering and notification in GDMO. This means interface specifications generated from a common OMT model will sub-optimal for implementation using either of the interface definition languages.

Significantly, the TMF has not yet integrated its business process model closely with requirements capture in this methodology, though this is an item of study within the Forum. In practice, the business process model is just used to scope a study area rather than in the analysis of requirements, e.g. [nmf-504][chen97b]. It would seem likely, however, that techniques such as EPCs or UML activity diagrams would be useful in determining how this common process model may be systematically refined into individual interface agreements. It is pointed out in [mcleod], for example, that grouping activities into vertical swim-lanes in UML activity diagrams can be used to identify the separate domains, typical in SMS problems. The boundaries between swim-lanes can therefore be used to determine the control and information flow requirements of interfaces between those domains. A similar approach to refining interface designs from the TMF Business Process Model is presented by the author in [lewis99a] and is assessed as part of Case Study 5 in Chapter 4.

Other OOAD methodologies have been applied to SMS development. In applying the FUSION OOAD methodology, it is noted in [saydam] that though the approach led to a straightforward and consistent design of a solution from an object oriented

analysis of customer needs, it was not well suited to the development of distributed system. Few OOAD techniques currently address the needs of distributed systems well though this is an area of ongoing investigation. For instance the OMG have recently released an RFP for a UML profile for CORBA [ad/99-03-11]. The application of OOAD techniques in the context of Open Distributed Processing is discussed in more detail in the next section. There are few reports of experiences of SMS development using CASE tools, but in [neilsen] an OMT graphical editing tool was found invaluable in speeding the generation of the various models and presenting and revising diagrams for discussions between developers for a VPN SMS. This work formed part of the project studied in Case Studies 1 and 2 described in the next chapter.

To summarise, OOAD techniques have much to offer to the development of SMS, and are also finding application in the development of standards at least within the TMF. Business process modelling is a potentially important technique in relating business requirements to system requirements, but techniques for integrating this with OOAD techniques are not well established. Finally, the use of design pattern could potentially be applied to the exchange of knowledge between SMS development stakeholders, but the immaturity of techniques to integrate patterns with OOAD models makes them difficult to include in any methodological guidelines.

3.2 Open Distributed Processing Related Methodologies

In response to some of the problems raised by the complexities of large scale distributed systems, ISO has developed the Open Distributed Processing Reference Model (ODP-RM) [x901][x902]. This reference model aims to support the specification of systems with the following properties:

- Openness in the portability of components between different processing nodes and in the interworking of components in different systems.

- Integration of various systems with heterogeneous architectural, resource, performance characteristics into a whole.
- Flexibility in supporting the evolution of systems and their components as well as run-time reconfiguration, such as when handling user mobility.
- Modularity to maximise the autonomy of interrelated components, which is required for flexibility.
- Federation of systems from different technical and administrative domains.
- Manageability of systems to support policies related to configuration, Quality of Service (QoS) and accounting, amongst others.
- Provision of Quality of Service in terms of timeliness, availability, reliability and fault tolerance.
- Security including authentication and access control facilities.

As well as supporting the definition of systems or components, ODP-RM is intended as a meta-standard intended to guide the development of other standards by providing a framework for the integrated support of distribution, inter-working, inter-operability and portability, all of which are relevant to service management. ODP approaches the problem of describing distributed systems by expressing the effects of distribution as a number of distribution transparencies, e.g. location, access or failure transparencies. These transparencies aim to hide the complexities of distributed systems from the software developer. Distributed systems adhering to the ODP framework are described using five complementary viewpoints of a system [x903]. These separate viewpoints together aim to provide a complete and consistent view of the system. These viewpoints are:

- The *Enterprise Viewpoint*, which aids requirements capture and is concerned with the business needs of the system in terms of its purpose, scope and policies.
- The *Information Viewpoint*, which aims to identify the information content of the system in terms of constraints on its use and its interpretation within the system.

- The *Computational Viewpoint*, which aims to describe the functional decomposition of the system into objects suitable for distribution.
- The *Engineering Viewpoint*, which addresses system support for distributed applications. It identifies the platform support needed to provide the distribution transparencies assumed in the computational view.
- The *Technological Viewpoint*, which addresses the basic hardware and software characteristics independently of the part they play in a specific distributed system.

Each viewpoint has an associated set of concepts and rules relevant to the concerns of that viewpoint [x904]. As viewpoints are separate but inter-related views of the same system, the relations between terms in different views are subject to consistency constraints. One aim of ODP is that viewpoint languages may be defined in a formal way that would enable the automation of consistency checks between viewpoints. Bindings have been suggested between ODP viewpoint concepts and Formal Description Techniques (FDTs), such as LOTOS and Z. Conformance in ODP may be expressed in terms of a reference point specified using combinations the viewpoints. An example reference point specification could contain: an enterprise specification giving roles and policies across the reference point; an information specification giving the universe of discourse for the reference point and a computational specification of the operations and dialogue across the reference point. The ODP-RM, however, does not specify any methodology describing how the viewpoint-based specifications should be developed. Approaches to establishing an ODP-based methodology have varied, from those who have attempted to developed the FDT approach to support automated translations between viewpoint and to those who have tried to apply semi-formal specifications to expressing viewpoint concepts. As pointed out in [erdmann], ODP FDT language mapping to Z and LOTOS are not suitable for use in the Enterprise viewpoint which must be used in requirements capture activities involving potential system users and domain experts who will not be conversant with such notations. In the author's opinion, the

FDT based approach is not well suited to any viewpoints in the context of SMS Development since, as established in Chapter 2, a suitable development framework must be easy for the range of practitioners to understand and use. However, the application of FDT's to telecommunication system development is analysed further in the next section. The rest of this section therefore only deals with the application of semi-formal techniques to ODP-based methodologies.

An important factor in considering the practical application of ODP is that no popular distributed processing platform conforming to ODP and associated ITU-T standards has emerged. Instead, partly inspired by ODP, the OMG's CORBA represents the de facto open distributed processing platform in use today. In this context Microsoft's COM [kindel], though it is widely used, is not considered a truly open distributed processing platform as its specification is under the control of a single company. This situation is changing however, with the recent release of potions of COM to the Open Group for the support on non-Microsoft operating systems. CORBA provides the equivalent of the distribution transparencies provided by the ODP engineering viewpoint concepts though a combination of the core ORB capabilities and CORBA services, though the structuring is different and the concepts and functional separations do not match precisely. Though OMG RFPs pay some lip service to the use of ODP principles and viewpoints, they are not used in practice for OMG standard development. Significantly, the OMG's standard OOAD notation, UML, does not refer to or explicitly support ODP viewpoint concepts. Attempts to apply ODP viewpoints in practice has therefore tended to focus on the use of the enterprise, informational and computational viewpoints during requirements capture, analysis and design, but with interoperable interfaces being ultimately expressed in the OMG's Interface Definition Language (IDL).

Before examining individual ODP-based methodology experiences, a better understanding of the ODP concepts in the enterprise, informational and computational viewpoints is required. The enterprise viewpoint support the following concepts, represented as enterprise objects and their relationships:

- *Environment:* Part of the model that interacts with, but is not part of, the system being specified.
- *Role:* The view of an entity in terms of an agent that carries out some activities with respect to another agent.
- *Resource:* An entity that is operated upon.
- *Contract:* The agreement of the activities and information that pass between two enterprise objects.
- *Policy*: Rules specifying the constraints and obligations a subject has towards some target.
- *Community:* A group of objects with a common objective.
- *Domain:* A collection of enterprise objects grouped for purposes of autonomy, authority and control, often driven by the structure of the business organisation under scrutiny.
- *Federations:* A community of domains grouped to serve some common objective.

The information viewpoint concepts are expressed in terms of Information Objects (IOs) and the static and dynamic relationships between them. They are therefore very similar to OOAD class and object modelling techniques such as those found in OMT. The computational viewpoint is based around the concept of a computational object (CO), which offers functionality through one or more well-defined interfaces. This model differs from the current CORBA model where an object has only one interface, though as observed in [kitson], ODP COs can be implemented by grouping CORBA interface objects.

Probably the largest body of work that attempts to apply ODP principles to telecommunications and implement the results using CORBA, is that of the TINA-C, which was introduced in Section 2.2.4. Though TINA adopted the use of ODP viewpoints, it has not adopted the viewpoint languages suggested in [x904]. For the

enterprise viewpoint, ODP concepts are condensed to statements of which stakeholder are involved, the roles they play, the services they offer and the obligations of service customers. The ODP concept of federation is used in the TINA business model [mulder]. For the information viewpoint, [christensen] advocates the use of the OMT object model for TINA. TINA supplements this with a textual notation consisting of quasi-GDMO object definitions and an object relationship model based on the OSI General Relationship Model [x725]. This notation provides a representation of objects, including their attributes, the constraints and operations that cause change in the object's state, as well as object inheritance and relationships between objects. For the computational viewpoint, TINA-C has adopted its own graphical notation consisting of simple component diagrams representing computational objects and the operational and stream interfaces they offer to each other [natarajan]. To provide detailed definitions of computational object interface structure and operation definitions a superset of IDL termed, Object Definition Language (ODL), is used [mercouroff95]. ODL allows the definition of multiple interfaces, of stream interfaces and of references to interfaces used on other objects. ODL enhances the ODP concept of a CO by supporting the grouping of COs into building blocks that can be manipulated as a unit for the purpose of system life-cycle management. For the engineering viewpoint, a Distributed Processing Environment (DPE) is assumed which provides various distribution transparencies required by COs through a set of services made available to engineering objects populating the DPE [graubmann]. The engineering objects themselves are arrived at directly by decomposing the computational objects (COs). As TINA has adopted CORBA as its DPE, this decomposition involved mapping ODL to IDL.

TINA goes beyond the ODP-RM by specifying an outline methodology for developing TINA services [salleros]. This methodology presents a development process where the enterprise viewpoint of a service is addressed during the analysis stage of the development process, followed by information viewpoint modelling and computational viewpoint modelling which together result in the system's design specification. The computational model and the derived engineering model both

support the implementation of the system. Though the information viewpoint model and computational viewpoint model are described as complementary parts of the design model, exactly how to iterate between them and to describe the relationship between the different objects in these models is not stated in any prescriptive manner. The different cardinalities that may exist for CO to IO mappings are described, but no guidelines are given for situations where the different relationships might best be applied. Also, though there is a relatively clear mapping from COs to engineering objects, no guidelines for the implementation of IOs is provided.

The TINA Management Architecture prescribes the use of the manager-agent paradigm for managing TINA resources through manipulating and monitoring managed objects. Though this approach is not in evidence in the service management specifications in the Service Architecture, it has been applied in implementation of elements of the TINA Network Architecture for ATM management [griffin97]. Here MOs were accessed via a management broker CORBA interface. This approach was regarded as more flexible and amenable to reuse in different applications than those based on higher-level, task-oriented interface definition, which were subject to change as new management task were identified. If we consider IOs in the management context to be MOs then this approach points to a more definitive way of identifying IO to CO mappings for SMS.

Though the TINA-C was concerned primarily with the development of specifications, it has associated with it several auxiliary projects that developed TINA-based systems and extended its specification base. These projects, therefore, provide a further source of experience on the application of ODP viewpoints. One of the major results of this work has been the evolution of ODL in describing the notation and semantics of both its graphical and textual variants [mercouroff97]. This has now been adopted within the ITU-T as a notation for computational objects [itu-odl] and has also helped influence the OMG towards the investigation of multi-interface objects [omg/96-01-04].

The computational object is also seen as the major unit of reusability in TINA, with ODL allowing the description of a CO to be partitioned into the interfaces for the services it provides and those used for the management of those services. It also allows for a CO definition to include the identification of the services of other COs that it uses, thus providing a more complete description of the CO for reuse. The ODL meta-model is adapted and extended in [dede] to provide behaviour descriptions for COs using Service Definition Language (SDL) [z100], which is discussed further in the next section. The author was involved in an attempt to apply ODP-viewpoint based models to the development and deployment of service management components [lewis97]. Here the structuring of the computational model, as building blocks was found to be effective in reusing service management components in different business scenario. However, as described in Case Study 3 and in [wade97], the use of both the information and the computational models for describing the design of systems was problematic.

Some prototype tool support has been generated specifically for TINA modelling purposes [bosco]. Primarily, this provided graphical and textual editing for computational modelling using ODL for object and building block definitions, together with object behaviour in SDL. This tool was combined with IDL and C++ generators for integration into simulation and testing tools. Such CASE-based development activities are addressed further in the next section in relation to SDL-based development.

An early attempt at applying ODP viewpoint to a service management development framework was conducted between 1992 and 1995 by the EU funded project PRISM [berquist]. This provided early validation of TINA modelling techniques as well as borrowing concepts from it. This project developed, through a set of paper-based case studies, a structured development methodology based on ODP viewpoints but with more detail than given in TINA documentation. It also possessed a more explicit linkage to the TMN interface definition methodology, M.3020. The overall PRISM methodology is summarised in Figure 3-1 using a notation similar to UML activity diagrams.

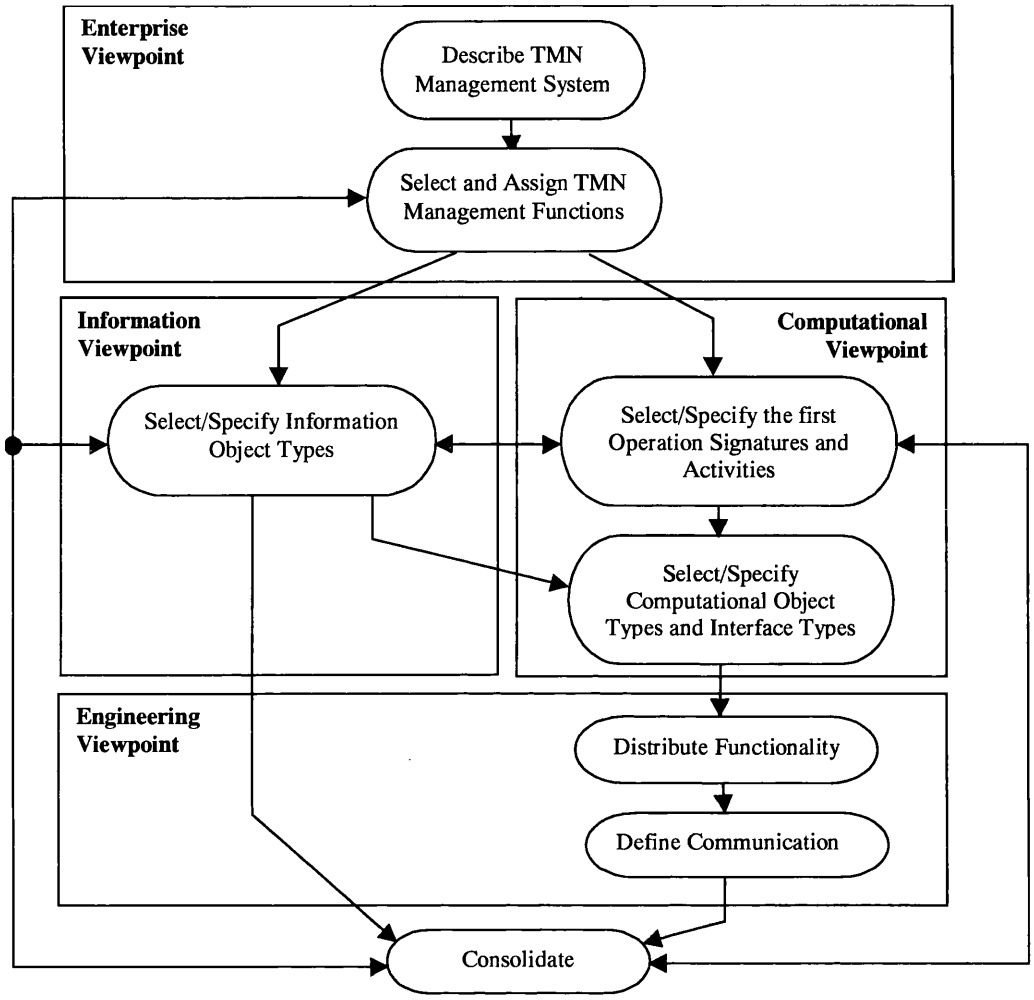


Figure 3-1: The PRISM, ODP viewpoint-based development process

The process for developing the enterprise viewpoint for a specific system involved the following sequence of tasks:

- Identifying the Environment and Parties Involved: This introduced the idea of an actor, which could play several roles with respect to other actors. Relationships between actors were identified with specific goals and were used to identify QoS attributes, the duration of relationship, availability conditions and safety and security considerations for the relationship.

- Requirements Capture: Requirements imposed by individual roles are recorded and categorised by management functional areas (a set extended from FCAPS) and TMN logical layers.
- Description of Management Services: This defines management service required by users and decomposes them into management functions using the process defined in [m3020-95] as described in the next section.
- Structuring the Enterprise: This refines the identification of actors, roles, contracts, requirements and management services into a detailed, structured set of objectives. These are expressed as enterprise objects grouped using ODP concepts of community, federation and domains. Policies are formulated that define which activities a manager may perform (i.e. authorisation policy) and which ones they must perform (i.e. obligation policy).
- Scenario description: This describes the sequence of interactions that may occur across multiple domains. These express the interactions that may occur between actors and roles grouped by communities, federations and domains, in terms of Jacobsen-style use cases and as event-trace diagrams between actors.

The static relationships between the PRISM enterprise viewpoint concepts are given in Figure 3-2. This set of semantics for the enterprise viewpoint was adapted from the corresponding ODP set of concepts with the aim of supporting more directly the requirements of TMN systems [strick94].

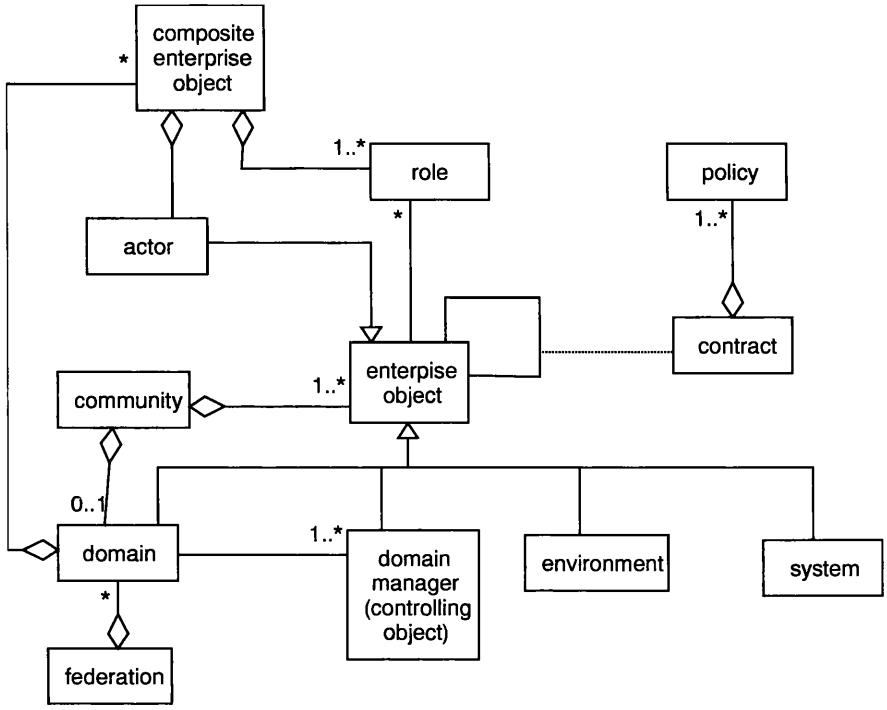


Figure 3-2: The PRISM Enterprise Viewpoint Concepts

For the information viewpoint the OMT object model is used together with quasi-GDMO in a manner similar to TINA. For the computational viewpoint the following steps are taken:

- Computational Object Identification: This uses an analysis of the enterprise model, and in particular of the management functions identified, to help identify units of functionality.
- Dynamic Computational Modelling: This uses computational activity diagrams designed by PRISM [dahle] to represent the sequence of management operations exchanged between COs for a specific management activity, triggered by the user's invocation of a management function. These diagrams are therefore similar in their semantics to UML collaboration diagrams.
- Computational Object Type Design: This uses Computational Object Type diagrams to represent the static relationships between COs and their attributes, interfaces and operation signatures. This is effectively a combination of OMT or

UML class operations (though allocated to multiple interfaces) and attribute notations from the graphical ODL representation.

- Building Block Design: This involves grouping computational objects into more course-grained systems that may correspond to TMN Operations Systems, according to criteria of release independence, security, system management, distribution or TMN logical layering. Building Blocks are represented as enclosing rectangles in computational object type diagrams.

As PRISM did not attempt to implement its designs or map them to CORBA, their approach to engineering modelling is not analysed here. As with TINA, the mapping between the different viewpoints did not seem to follow a very well-defined process, though some mappings were identified between objects in different viewpoints. For instance the methodology advises that the information model be based on an analysis of the text description of enterprise model use cases, with noun phrases mapping to IOs in the first instance. Also, CO operations should be provided for all management functions identified in the enterprise model, though this does not help in the design of CO operations that do not face the user. As with TINA, mapping between IOs and COs is identified as being potentially many to many. In [may] the need to iteratively regard both informational and computational models in order to gain complete and consistent specification for both is described. How the relationship between the information and computation viewpoints might guide the identification of COs is also not well addressed. It is acknowledged in [berquist], however, that this is a difficult task as it involves consistency checking of both static and dynamic models in both viewpoints. The grouping of COs into building blocks was also found to be problematic since the different grouping criteria often result in conflicting or overlapping groups. Another issue not fully addressed in the PRISM methodology is how the managed object definitions that specify the details of TMN management functions might be imported into ODP models.

Some tool support was developed for the PRISM methodology [strick96], which supported implementation and specification repositories and allowed grammar

modules to be provided for each viewpoint, acknowledging that the notations used would change over time.

Others have attempted to develop SMS using subsets of ODP viewpoints coupled with other development techniques. In [choi], a design for a service order handling interface is developed in terms of COs and IOs, but driven by a TMF-like business process model rather than from an enterprise viewpoint model. This approach also uses state transition diagrams in modelling IOs and links this to the CO model by binding triggering events to CO interface operations. The author was involved in an attempt to integrate a UML-based SMS development approach with a design expressed in terms of COs and IOs [lewis99d]. This is reviewed in more detail as part of Case Study 4 in the next chapter, with further examples of the application of this techniques given in [tiropanis98] and [hellemans99].

More recently, others have attempted to apply ODP to SMS development using UML [kande]. The object oriented nature of both UML and ODP, and the similarity of some UML models to models used for ODP viewpoints, made this approach fairly straightforward notationally. The viewpoints were represented using the following diagrams:

- Enterprise Viewpoint: Use case diagrams, class/package diagrams.
- Information Viewpoint: Class/package diagrams, state transition diagrams.
- Computational Viewpoint: Sequence, collaboration, component, activity and class/package diagrams.
- Engineering Viewpoint: Component and deployment diagrams.

In this study, using a single notation (i,e, UML), edited within a single CASE tool, proved useful in making the models for the different viewpoints more coherent and easier to navigate between. Several ODP concepts, such as enterprise objects, could not be directly represented in UML, but were readily modelled using stereotype of the class type. Limitations were found however in describing COs, as not all aspects of ODL could be conveniently represented in UML.

The ITU-T has a standardisation activity developing an Open Distributed Management Architecture (ODMA) [x708], which is applying ODP principle to OSI Systems Management. The draft output of ODMA to date has focussed on how CMIS-based manager agent systems can be represented in the ODP engineering viewpoint, though for practical purposes this work has been superseded by the JIDM CORBA-CMIP gateway specifications.

3.3 Telecommunications Specific Methodologies

This section reviews relevant development methodologies that originated in the telecommunications industry. This includes ones that combine elements of general software engineering techniques and ODP-based techniques and ones that apply specialised techniques to SMS development and related areas such as network management and service control system development.

As mentioned in Chapter 2, the TMN set of recommendations contains a recommendation, M.3020, for an Interface Specification Methodology [m3020-95]. This provides guidelines for the functional decomposition of management interfaces found at reference points in the TMN functional architecture. This methodology is primarily aimed at guiding work conducted in ITU-T working groups towards the standardisation of management services [m3200] and management functions [m3400], though application specifiers and protocol specifiers are targeted also. In M.3020, management services describe the functionality available at a TMN interface from the point of view of the user. These services are decomposed into management functions, using either existing management functions or defining new ones if required. Management functions are grouped into management function sets for the purposes of information modelling. Where possible, management functions are based on OSI system management functions. Information modelling involves analysing existing generic and technology specific information models to see if existing object classes satisfy management function requirements. If existing management functions have been used, then these may have corresponding information models already defined which then can be reused. Though it is expected

that TMN information models will be accessed over the interface using CMIP, the development of other protocols is also accommodated by the methodology. The overall process as represented in [m3020-95] is summarised in Figure 3-3.

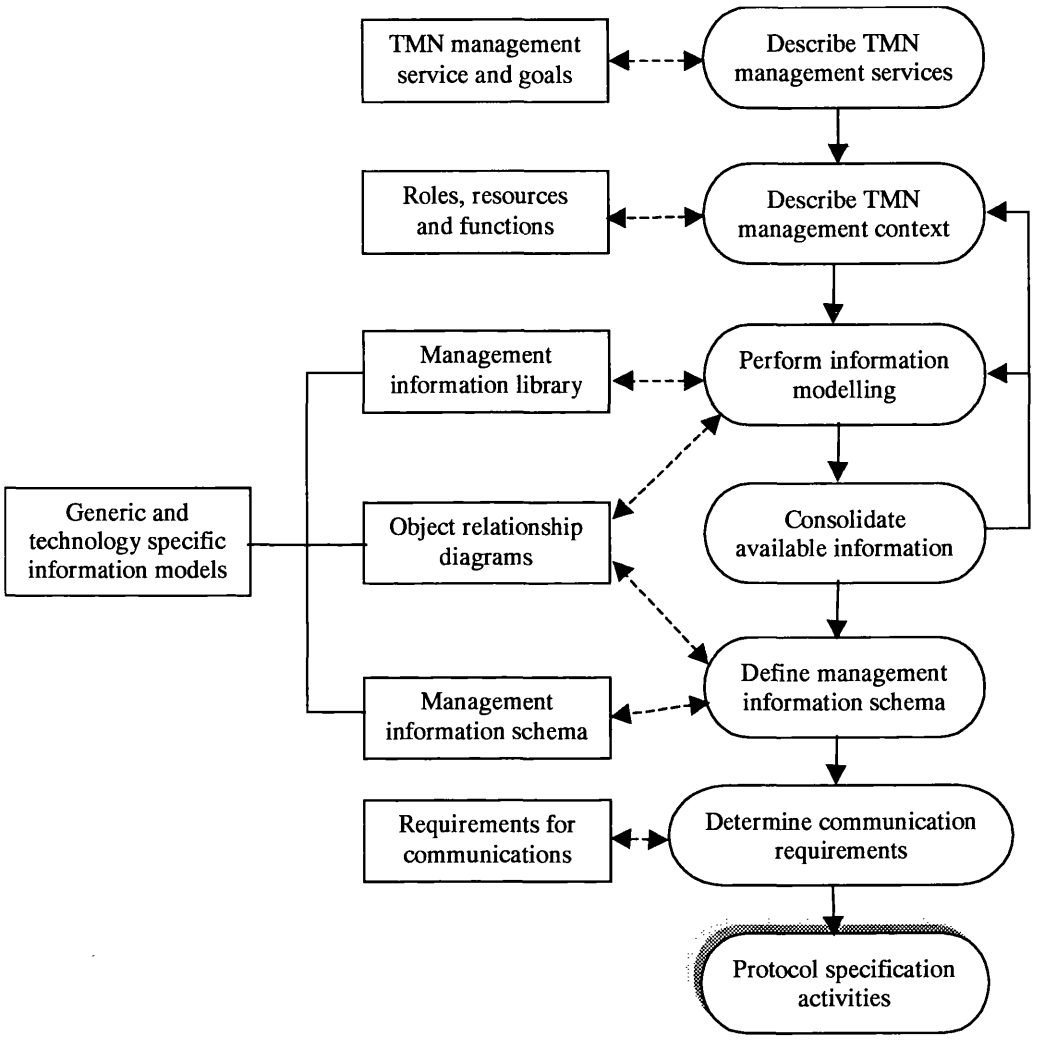


Figure 3-3: The M.3020 development methodology

The description of management services includes the description of the goal of the user in invoking the service. This is complimented by the description of the context in which the service is invoked, described in terms of the role played by the user, the telecommunications resources being managed and the management functions used, supplemented by scenarios descriptions giving examples of the functions' usage in performing the service. The service is also identified with respect to TMN interface

types, e.g. Q, F, X. The information modelling activity can be enhanced by the use of entity relationship diagrams, and must also express the object class naming schema that will be visible at the interface.

This methodology uses functional decomposition in order to subdivide the problem of interface definition and to support the identification of existing functions and information model that can be reused. It therefore offers no opportunity to specialise existing functions in a structured manner, with this only being possible using object class inheritance at the information modelling stage. Also this methodology does not provide any direct guidance on the functional decomposition of systems, only of interfaces. These characteristics represent a fundamental difference between the M.3020 approach and the general software engineering and ODP-based development methodologies described in the previous two sections.

The problem of functional system design was addressed directly in [griffin96] where the M.3020 was extended to include the identification of OSFs. Observing that the application of M.3020 directly to TMN OS development would result in single, monolithic OS in each logical layer, this approach focuses on the development of managed object clusters, which presented a single management interface, but which are also defined by their use of management functions from peer or subordinate clusters. These MO clusters represented the building blocks from which OSFs were built. The application of the methodology across a TMN requires the nomination of a system designer who maintains a consistent view of how OSFs were to interact with each other. The system designer therefore guides and co-ordinates the developers of multiple, individual OSFs, who follow M.3020 in defining their own interface information models. This resulted in a set of service and network management OSFs consisting of well-defined management functional components implemented as managed object clusters and associated manager functions [griffin95]. Another example of TMN interface specification development [covaci] show that techniques such as message sequence flow diagrams are useful for showing how management functions are used in specific scenarios and state transition diagrams are useful for defining the changes of MO state.

The TMF built on TMN standards to provide the additional industry agreements on management interfaces needed by its membership. It quickly realised that the structure of TMN standards, resulting from the application of M.3020 within the ITU-T was not easily usable by its member when used to implement and verify TMN systems. The separation of management service and function overviews from individual management function definitions sometime made it difficult to understand the mode of application intended for a function definition and associated information model. As the definition CMIS-based manager-agent interfaces allows very flexible use of the interface, this led to misunderstandings about the exact operation of an interface, which that had a negative impact on interoperability. The TMF (or the NMF as it was then) addressed this by publishing interface agreements in the form of an *ensemble* [nmf-025]. An ensemble packages together; an outline of what is to be managed, expressed as resources; what functions are required to solve the management problem and some scenarios to illuminate how these functions should operate dynamically on the resources. The process for developing these ensembles largely revolves around the identification of management functions and MO definitions from existing standards, with new MOs being defined only when absolutely necessary. The ensemble form differs from M.3020 in that it packages the motivation for the interface and conformance test specifications with the specification of the interface itself. It does not, however, provide scope for new management protocols to be defined, relying instead on the use of existing ones, e.g. CMIP. It also encourages the use of entity relationship diagrams and sequence charts of CMIP interactions to make the specification more accessible to its users. The ensemble approach was found useful in [bleakley] for defining TMN X-interfaces for accounting management, though it was observed that the techniques could be improved by the application of detailed use cases descriptions and a clearer mapping between management function and CMIP interactions. The ensemble form is no longer directly used in the TMF as the framework of which it was a part [zeisler] has evolved into the business process based framework used today. However, the legacy

of closely linking requirements for interfaces with their standardised specification still persists in the TMF development approach.

The EURESCOM organisation, which is funded by European public network operators to perform telecommunication related research, has conducted case studies into TMN development with the aim of providing guidelines for the development of TMN systems. In 1996 the EURESCOM project P414 used a paper-base case study of VPN service management to assess the possible merging of the TMN interfaces specification techniques, namely TMF ensembles and the M.3020, with OOAD techniques such as OMT, OOSE and FUSION. A report on this case study [p414-d2] makes several initial observations. It singles out use cases as a techniques that was found useful in bridging the gap between TMN interface specification approaches and OOAD techniques. Comparing the ensemble approach to M.3020, it concluded that the former was better suited to inter-OS interface definition, i.e. X or Q interfaces, while M.3020 was better suited to WS to OS interface definition, i.e. the F interface. A more detailed analysis of this case study [p414-d3] that attempted to combine the ensemble approach with OMT points out the key difference between the former as an interface specification technique and the latter as an application development technique. It recommended that the understandability of ensembles could be improved by employing the graphical notations of OMT. However it concluded that the ensemble process did map well onto the OMT process as the iterative and incremental nature of the latter was not reflected in the waterfall structure of the former. It therefore recommended that though the structure of ensembles aided understanding of the interface specification, to gain the full benefit of OOAD techniques such as OMT, this form should only be used for the final structure of the specification while native OOAD models should be used to perform the development process.

In 1997, another EURESCOM project, P.610, reported on its analysis of the state-of-the-art in development frameworks and methodologies for the management of multimedia services [p610-d1]. After analysing the methodologies used in P414, PRISM, the EU-funded Prospect project (the subject of Case Studies 3 and 4 of the

next chapter), and TINA, it concluded that as the de facto standards OOAD modelling notation UML should be adopted for any development methodology in this domain, with OMT providing guidance on the process. The results of several paper case studies were presented in [p610-d2], which suggests a UML-based methodology for developing multi-media service management systems that consists of the following steps:

- Requirements Capture: This consists of describing the service, providing a business model of the service, modelling the relationships between actors to determine the resources and actions required by the system (following the ORDIT methodology [strens][dobson]), describing use cases and the defining of scenarios, i.e. use case instances.
- Object Oriented Analysis: This consists of a domain track and an application track. The domain track involves building a class model of the problem domain. The application track involves building an application class model and refining it though the construction of sequence, state and collaboration diagrams.
- Mapping to Multi-Domain Management Architecture: This consists of grouping classes into packages according to functional criteria. This grouping should also aid in the identification of reusable packages.

No practical assessment or experience is relayed, however, from the execution of these case studies.

Other, more specialised, development techniques have emerged from the telecommunication software sector that may prove applicable to SMS development. Two approaches that have been applied in a variety of forms to network and systems management are policy-based management and formal managed object behaviour definition. Policy based management, as described in [wies], involved analysing high-level corporate policies and refining them into task oriented policies. Task oriented policies are then mapped onto functional policies that act on management services which in turn act on low-level policies that restrict the behaviour of managed object classes. Policy based management has been applied mostly in

situations where manager applications have to interact with potentially large numbers of network element or system management agents, e.g. [putter]. Its benefits come from enabling management applications to manage groups or domains of MOs [alpers][sloman], which may be distributed over potentially large numbers of agents, in terms of goal-driven policies rather than of specific management operations. An architecture is proposed in [davidson99b] in which network management systems are developed and sold as independent functions enforcing policies. Though this matches some of the SMS development requirements by explicitly supporting a market in management components, it is not clear how this would translate from network management to the service management environment. Policy-based management is largely focussed on dealing with problems in a single layer of manager-agent relationships, so its application to the multi-layer TMN architecture, from which the definition of service management used here is derived, is also not clear.

Formal MO behaviour has been an ongoing area of study with the ITU-T, motivated by the problem of ensuring interoperability between management systems based on GDMO definitions that provide only natural language behaviour descriptions. A wide range FDTs seem to have been proposed for this task including SDL [carls][barbeau], LOBSTERS [festor], DOMAINS [fink], Object-Z and RAISE [derrick]. However, no agreement has yet emerged on a common language for formal MO behaviour description, and these techniques suffer from a lack of integration with common CASE tools, i.e. those supporting OOAD methods.

SDL has been used widely in the development of telecommunication control systems and intelligent networks [morris][olsen99]. It allows for the analysis of specifications for correctness, for validation through simulation, for generation of implementation code and the generation of test cases and test code. The development of notational mapping between SDL and both IDL [olsen95] and ODL has allowed SDL to be used for such development activities in TINA-based service system development [lucidi][schieferdecker]. The feasibility of this implies that SDL could also be applied to such activities in TINA-based SMS. However, as pointed out in

[lodge], many of these techniques rely on an existing, well defined and relatively narrow functional frameworks within which SDL based activities are cost effective. As discussed in Chapter 2, no such well-defined framework exists for service management, thus limiting the extent to which these techniques can be usefully applied to SMS development.

Another complimentary approach to developing telecommunication systems is one that focuses on integrity. Integrity is the ability of a system to retain its specified attributes in terms of performance and functionality, and is important for telecommunication systems due their increasing complexity, especially for signalling systems. The management of integrity involves the prediction of where a system's design has high integrity risk areas, testing to validate integrity of a developed system and maintenance to monitor and diagnose threat to integrity in operation. As suggested in [monton], the prediction activity requires modifications to existing development methodologies so that integrity risk assessment, possibly based on complexity metrics, is introduced at each stage of development. This technique has been applied to a multi-domain SMS case study [prnjat], where complexity metrics were applied to ODP viewpoint models of the system generated using UML.

3.4 Summary of State of the Art Analysis

From the analysis of methodological techniques applied to service management, the author concludes that the area of development frameworks and methodologies for problem domain has not been extensively investigated and the level of experimental rigour adopted is low. Most reports consist of assertions made by their authors that were based on their own experiences of using a technique. Only a few reported case studies conducted on the application of a technique in a wider project or provided the lessons leant from such projects. The author found no quantitative assessments of the application of methodological techniques to SMS development. This, however, is unremarkable when compared to the main body of work into methodological techniques for software development. In a survey of over 600 software engineering publications [zelkowitz], the majority contained no experimentation or resorted to

assertions, only 10% presented case studies, while other, more controlled experimental techniques presented in only a very small percentage of the sample. Of the development approaches presented in the previous sections that were assessed in more detail, most were based on paper case studies that only extended as far as a system design specification. The minimum, implicit quality check that results from observing if the design led to a working implementation within reasonable costs constraints was therefore largely absent.

Based on the qualitative material available it can be observed that telecommunications management development techniques borrow heavily from software engineering techniques popular at the time. It can therefore be concluded that the growing popularity of UML, its establishment as the de facto OOAD modelling notation, its widespread tool support and its expanding skill base will make it a very acceptable choice as the notation for an open SMS development framework. Other OOAD techniques that seem to have gained widespread acceptability in SMS development are use cases, graphical class modelling and the use of sequence diagrams. The application of use cases seems particular suited to SMS development as such systems are essentially intended to support the activities of human operators and service customers, the analysis of such activities being the focus of use cases.

Other more specialised techniques seem to be driven by their use in standards bodies rather than clear results concerning their utility in industrial applications. Prime examples are the use of business process modelling promoted by the TMF and of ODP viewpoints advocated by TINA-C. However, as these bodies represent the major current source of service management standards, these approaches must be accommodated in addressing the needs of the SMS development stakeholder model. The ITU-T does not emerge as a very suitable source of methodological techniques. The functional decomposition approach taken by M.3020 has not yet been well integrated with OOAD-based techniques, and in the author's opinion needs to be reassessed as the basis for developing TMN interface specifications. The ITU-T's other methodological efforts have focussed mostly on FDTs, which due to their

specialised mathematical nature and the immaturity of accompanying tools, are not likely candidates for an open SMS development framework (see Goal 9 of the previous chapter). In the author's opinion, the same reasons for the unsuitability of FDT's also diminish the motivation for applying ODP viewpoints within an SMS development framework. Other, goal driven development techniques such as policy based management and integrity analysis, are complimentary to the application of more conventional techniques, as they provide guidance in the structuring of a solution. However, they do not provide techniques for performing the detailed modelling. Once techniques have been established that address the primary business needs of the SMS development stakeholders, these techniques may build upon them to allow the stakeholders to interact at a higher level of abstraction, for instance by selecting components by the goals they achieve rather than the functions they perform. As this thesis primarily addresses these business needs, the application of goal driven techniques is left for further study.

An important distinction that is apparent in analysing the various methodologies is the differences in approach observed when the target is an interface specification rather than a system implementation. As observed in [sullivan], the only techniques that would seem well suited to both is ODP viewpoint, though as pointed out in [schoo], when applied in TINA, the benefits of CO modularity and reuse in system design are not fully exploited in the definition of TINA reference points. A common development framework that addresses all the SMS development stakeholders must therefore address both goals in its methodological guidelines. Finally, though many of the techniques reviewed claim to support and ease software reuse, there is very little evidence of reuse of software or specifications between projects or between separate stakeholders.

3.5 Synthesis of Methodological Requirements for SMS Development Stakeholders

This section analyses the requirements for the methodological guidelines portion of a possible development framework, based on the state of the art review in the

preceding sections of this chapter. It does so by analysing the development processes that each of the SMS development stakeholder will undergo in the course of their core business activities. For the SMS Developer this process, is the development of an SMS, for the Software Vendor it is the development of a commercial off the shelf (COTS) software and for the Standard Developer it is the development of an interoperability standard. The stakeholders will also perform other activities such as market surveys, sales and internal guideline development, but for the purpose of analysing the requirements for a suitable development framework, only the core business activities are considered.

An initial assumption made about the structure of the development framework's methodological guidelines is that they will be split into the following parts:

- *Notations and Meta-model*: Providing guidance on the types of notations that are suitable for different development tasks and the structure of the models, i.e. the meta-model, that is appropriate for these tasks.
- *Process Guidelines*: Providing guidance on the specific activities involved in the overall development process, the relationships between activities and their relationship to notations and meta-models.

The overall generic structure of a development framework can therefore be modified from that shown in Chapter 1, to the one shown in Figure 3-4.

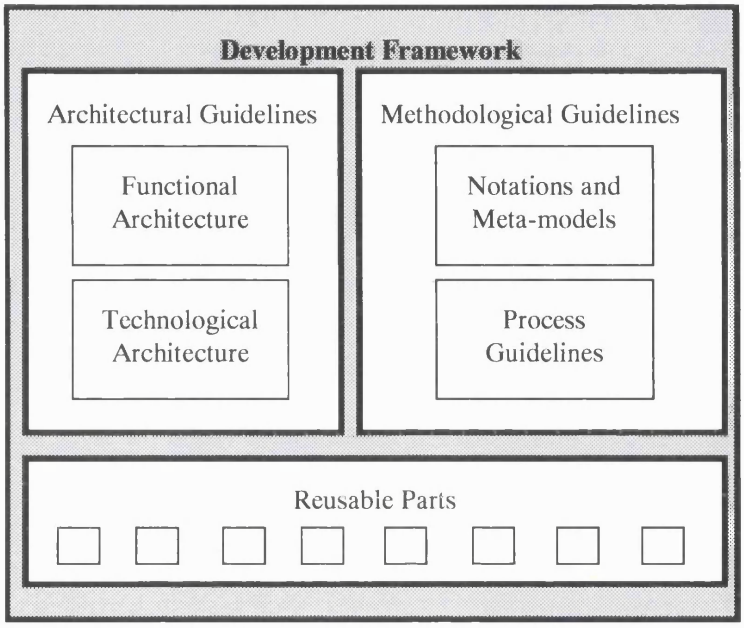


Figure 3-4: Refined Generic Development Framework Structure

This structure of methodological guidelines mirrors that taken by the UML authors and the OMG analysis and design working group in separating the notation and modelling semantics that make up the UML from the analysis and design process guidelines. The motivation for such a split is that notations and meta-models can usually be well-defined while development processes are a lot more difficult to capture since they are highly conditional on the context within which the development is occurring. For this reason notations and meta-models are often defined formally or semi-formally using existing (or sometimes their own) notation, while process guidelines are delivered as more general descriptive advice. This split is reflected in most of the methodologies reviewed earlier in this chapter, as summarised in Table 3-1. This table also includes the separations between functional and technological architectures and examples of the reusable parts that exist in the different development frameworks.

Development Framework/	**Methodological Guidelines**		**Architectural Guidelines**		**Reusable Parts**
Standards set	Notations and Meta-models	Process Guidelines	Functional Architecture	Technological Architecture	
TMN	GDMO	M.3020	M.3010	CMIP/CMIS	Management functions/ MOs
TMF	UML/OMT		Telecoms Operations Map	Technology Integration Map	Protocol Independent Models
TINA	ODL, OMT	Design Guidelines	Business Model and RPs	Engineering model (DPE)	
OMG	UML Notation and Semantics	OOAD Process RFP	OMA	CORBA/IIOP	CORBA Services and Facilities

Table 3-1: Categorisation of management related standards by generic development framework structure

The following sections present generic models of the core business activities of each of the SMS developer stakeholders. The graphical notation used is similar to UML activity diagrams, where lozenge shapes represent discrete activities, arrows between activities represent the sequence of activities over time and oblongs represent information that is used by or generated by activities. Round edged oblongs

represent stakeholder types, and may contain the information maintained and the activities conducted by that stakeholder. Information not contained within a stakeholder oblong is typically generated to support the exchange of information between stakeholders for a particular activity. A shadowed oblong indicates that multiple instances of the stakeholder type or of the information represented are involved in the overall process.

The aim of analysing these processes is to identify the commonalties in the development process experienced by the different development stakeholders in terms of the relationships between the activities they conduct and the information exchanged between activities and between activities and other stakeholders. As Booch points out, in any effective OOAD process the individual developers will iterate through the various development activities several times as the information exchanged is revised based on the experiences and insights obtained through conducting the previous iteration. It is assumed however that iterations will involve the same information being passed between the activities, and therefore the inclusion of activity iterations will not add much to the process analysis. The iteration through sequences or sub-sequences of activities is not explicitly analysed in the following sections, though it is the intention that the resulting development framework should support an iterative and incremental development process.

3.5.1 The SMS Development Process

A simplified depiction of the SMS development process within an idealised SMS Developer stakeholder is given in Figure 3-5. It is assumed in this model that the SMS Developer will maintain its own internal architectural guidelines. These may be purely proprietary, they may be influenced by architectures in the public domain, they may reflect architectures used by Service Providers that are consumers SMSs, they may explicitly conform to standard architectures or they may embody combination of these influences. It is also assumed that the SMS Developer possesses an existing set of products on which it intends to build when developing new SMS products. Ideally these existing products will conform to the internal

architectural guidelines. Examining each activity in turn we can identify the interactions on each one in more detail:

- *SMS Requirements Capture:* An SMS is primarily developed for a single customer, i.e. for a single Service Provider. Though an independent SMS Developer may endeavour to reuse some or all of an existing product in future SMS development contracts, each SMS development project will be primarily driven by the requirements of a single Service Provider, expressed in a requirements statement document. If the SMS is required to interact with the SMS of the Service Provider's customers or of Third Party Service Providers, then the requirements of these stakeholders may also need to be explicitly obtained by the SMS Developer. The result of the SMS Requirements Capture activity is the SMS Requirements Statement, typically expressed as a categorised set of plain language statements.

- *SMS Requirements Analysis:* This activity involves analysing the SMS Requirements Statement with the aim of synthesising a structured, logical model of the target SMS. This will be conducted at a high level of abstraction, largely ignoring implementation issues such as performance tuning and the choice of communications protocols or distribution technology, though still having to be aware of them where they have a direct impact on the logical structure of a solution. The Requirements Analysis will have to take into account the internal architectural guidelines both to guide the functional structuring of a potential solution and to use the technology architecture to understand the impact of the technology choices. The Requirement Statement may specify certain open standards are to be used in the solution, or the SMS Developer may opt to use some standards to ease future interoperability problems or to make future solutions more marketable. In this case the Requirements Analysis will be influenced by architectural guidelines, potentially from several different Standards Developers. The SMS Developer may also aim to use COTS software from one or more Software Vendors. In this case the Requirements Analysis must take into account the architectural guidelines in which the off the shelf

components are presented. These guidelines may be themselves based on open standards or they could be closely aligned with the SMS Developers own architectural guidelines. The different vendors' architectural guidelines may, however, be quite different, in which case they must be considered carefully in this activity to ensure that the integration of the COTS software is successful. The output of this activity will be the SMS Analysis Model.

- *SMS Design:* This activity is driven primarily by the SMS Analysis Model. Based on the logical structures contained within this, a detailed design of the system is performed, specifying software modules, functional units and interface definitions. This will require reference to the internal architectural guidelines and potentially also to those from Standards Developers and Software Vendors as referenced in the SMS Analysis Model. In particular, the technology guidelines will be examined in order to determine the impact on the design of the various computing and communication platforms used both internally and by COTS software, and of the need to support open platform interoperability. In addition, the details of the APIs and interface definitions via which existing SMS Developer products, existing Service Provider systems and COTS software products will be integrated will need to be considered. The definitions of any open interfaces the system conforms to will also need to be obtained from Standard Developers. The output of this activity will be the SMS Design Model
- *SMS Implementation:* This activity involves developing software code based on the SMS Design Model. Implementers may need to reference details of the architectural guidelines from the SMS Developer, the Software Vendors and the Standards Developers that are referenced in the SMS Design Model, though this should primarily be to refer to details in the technology architecture related to implementation. Issues presented by the functional architecture would already be embodied in the design and should not need to be referenced directly by implementers. In addition, implementers will need to follow references to the API and interface definitions of existing internal products, existing Service Provider systems, COTS software and open interfaces given in the SMS Design

Model. The construction of the SMS software may require the inclusion of modules from existing products and of libraries and executables from the Software Vendor. The resulting integration may require modification of the existing product modules and of the Service Provider's existing systems. In some circumstances the modification of the COTS products, either by the Software Vendor or if source code is available, by the SMS developer. The result of this software should be a set of SMS software modules for testing.

- *SMS Testing:* This activity will typically be highly integrated with the SMS Implementation activity, operating on the software generated by that activity. Testers will use the SMS Design and the SMS Requirements Statement as the basis for generating test cases and test harnesses. The APIs and interface definitions of existing products, existing Service Provider systems and COTS software will also be used in building test harnesses, while the open interface definition from the Standards Developers will be used to test conformance. Obviously, testing will be facilitated if the existing APIs and interfaces are accompanied by test environments and similarly if the open interface definitions are accompanied by explicit conformance statements. The output of the SMS testing should be SMS Software ready for deployment.
- *SMS Deployment:* This activity actually takes place in the Service Provider domain and involves ensuring that the SMS operates correctly with the Service Provider's existing SMS and NMS. This would typically involve the collaboration of SMS Developer staff and Service Provider operations staff.

This process analysis does not address the situation where two or more SMS Developers are required to collaborate in order to develop SMS in parallel for the same Service Providers, or possibly for two collaborating Service Providers. Replicating the Software Vendor to SMS Developer relationships in both directions could approximate the process relationships for such situations. This would reflect the need to jointly develop models at the different stages of development.

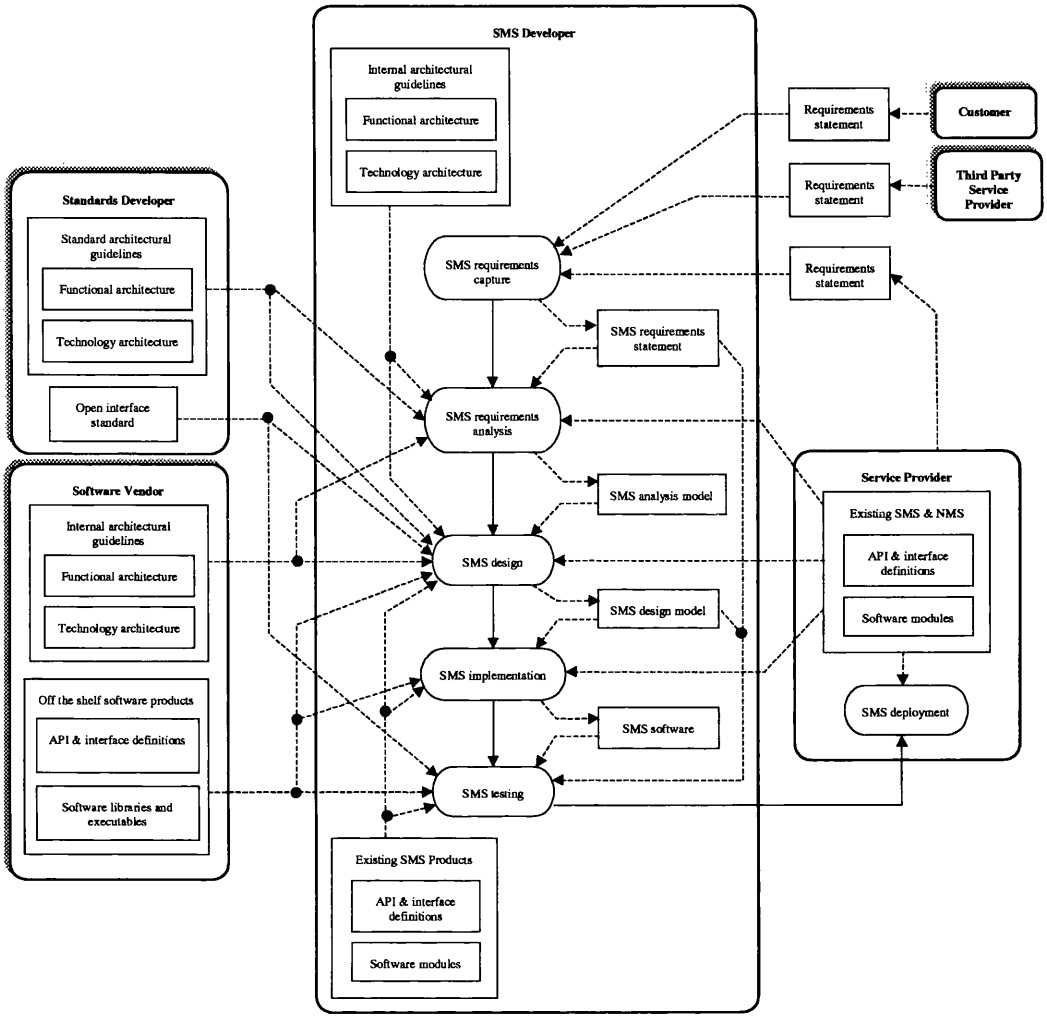


Figure 3-5: Process Model for SMS Development

3.5.2 The COTS Software Product Development Process

The process model for COTS software product development by the Software Vendor bears many similarities to that for the development of SMS software, as both are basically software product development processes. The following description of activities is therefore restricted to identifying where the process differs significantly from the SMS development process. It should be noted that, though COTS software may often incorporate other vendor's products, the interaction between Software Vendors is not analysed here.

COTS product requirements capture differs from the requirements capture activity for SMS in the source of requirements. Instead of getting requirements from a single customer and their collaborators, the Software Vendor needs to analyse the requirements of the range of organisations playing the SMS Developer role in order to identify a product with the widest or most profitable marketability. Requirements capture in this context therefore involves understanding the needs of SMS Developers, either through direct contact or through market surveys.

COTS product requirements analysis, design, implementation and testing do not differ greatly from the corresponding activities in the SMS Development process, except in that the dependency on the customer is greatly diminished. The analysis and design will only consider the existing systems used by SMS Developers in a general way, and there will not be an obligation to support products that will integrate with systems developed by a particular SMS Developer. Also, given that the interactions between Software Vendors is not analysed here, implementation and testing will not required direct integration with systems from another stakeholder. It is expected, however, that the dependency on architectural guidelines and open interfaces from the Standards Developers will be much more important for COTS software development than for SMS development. In other words, the COTS products will offer conformance to standards rather than necessarily guarantees of interoperability with other products. This is because off the shelf software developers rely more on the conformance of their products to standards to ensure wide marketability of its products. This reduces the risk involved in investing in COTS software development before any specific customers have been identified. The final activity shown is simply expressed as product release, since product deployment and use by a customer typically will not involve the close collaboration and testing in situ that would be required for a bespoke SMS product. The overall process model for COTS software development is summarised in Figure 3-6.

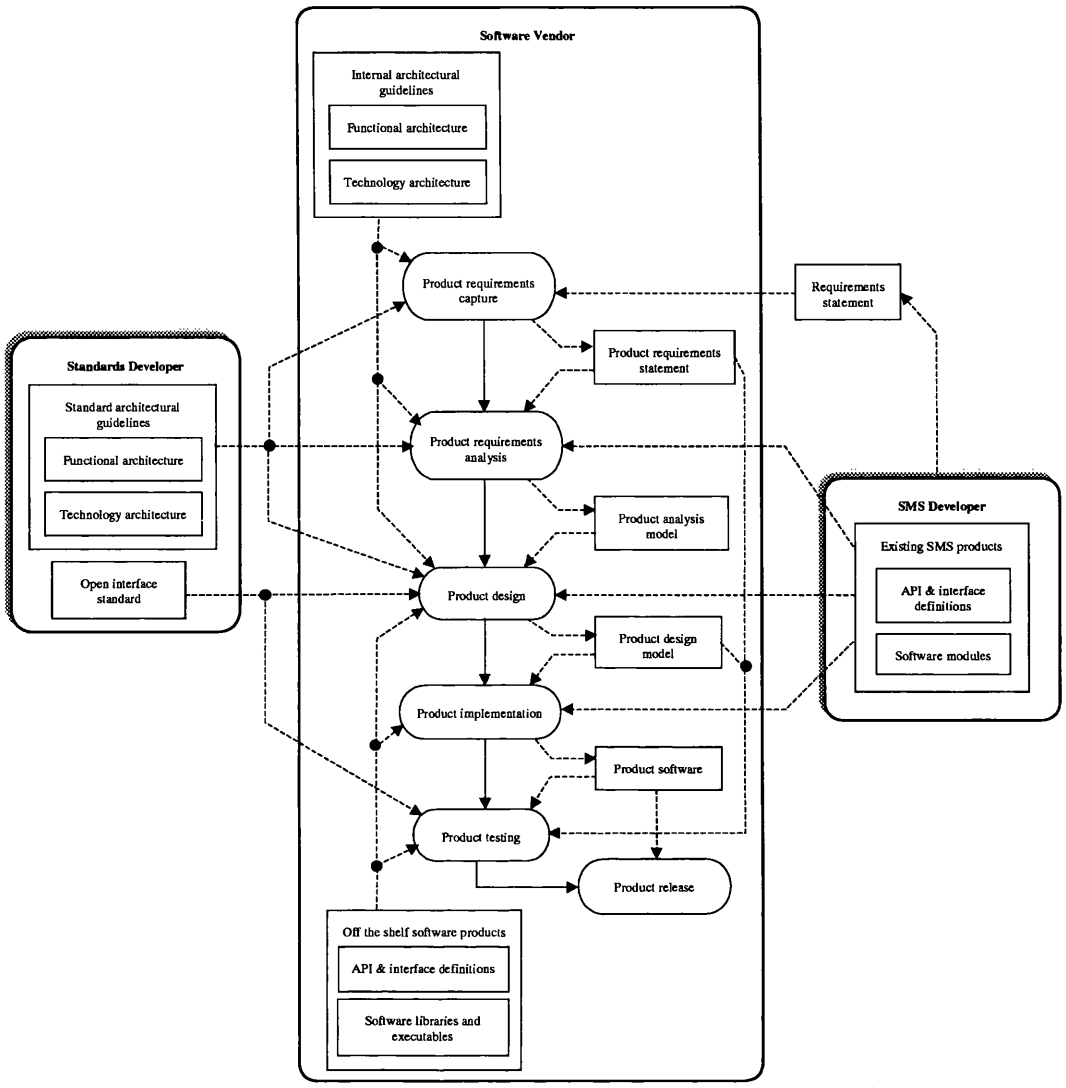


Figure 3-6: Process model for off the shelf management software development.

3.5.3 The Interface Standard Development Process

The interface standard development process differs fundamentally from the SMS and COTS product development process in that it does not result in the development of software, but only in specification documents. However, as observed in Section 3.4, the process of generating open specification does have some similarities with the software development process.

Capturing industry requirements for a new interface standard is a much more broad ranging process than capturing requirements for developing a product. The process may be visible externally, for instance as in the publications of Request for Information in the OMG. Alternatively it may occur within the standards body, with groups of members forming a consensus on which new areas need to be addressed. In the SMS development context, we would expect Software Vendors, SMS Developers and Service Provider all to be active in this process, as is reflected by the makeup of bodies such as TINA-C and the TMF. The Service Provider will be motivated to ensure that new standards are compatible with existing deployed NMS and SMS systems. The process used to define an interface standard varies between the different standards bodies in terms of its external visibility, and in the stages undertaken. Usually, however, it will be kicked off, after some analysis of the range of relevant requirements, by a clear statement of the problem being addressed and of the constraints on the solutions sought. This may be expressed as a Request For Proposals (RFP) or a Project Statement, depending on the openness of the development activity to external contributions. The design of the standard typically will involve the definition of an interface, some description of its behaviour and possibly some guidelines for assessing conformance of implementations. Once the interface definition has been accepted as satisfying the requirements and the constraints of the RFP, it may then join the set of interface standards maintained by the body.

An important difference between the analysis and design phases of a standard and those of a product is in the emphasis placed on consistency with existing standards. A key aim of standards is to provide a framework for software development that is consistent. There is, therefore, a great emphasis on ensuring that standards comply with existing standard architectural guidelines and that they are compatible with preceding standards from the same body. In addition, there is an increasing emphasis on ensuring consistency between standards offered by different bodies, particularly in avoiding generating a new solution to a problem that might be solved by an open solution from another standards body. Therefore the links between the analysis and

design activities and the architectural guidelines and open solution of the same and of separate bodies are strong. A summary of the process for developing service management interface standards is shown in Figure 3-7. This process model does not address the development of the architecture used by a Standards Developer. This is because this is typically a one-off process rather than an on-going one, and therefore not likely to benefit from a common methodological approach.

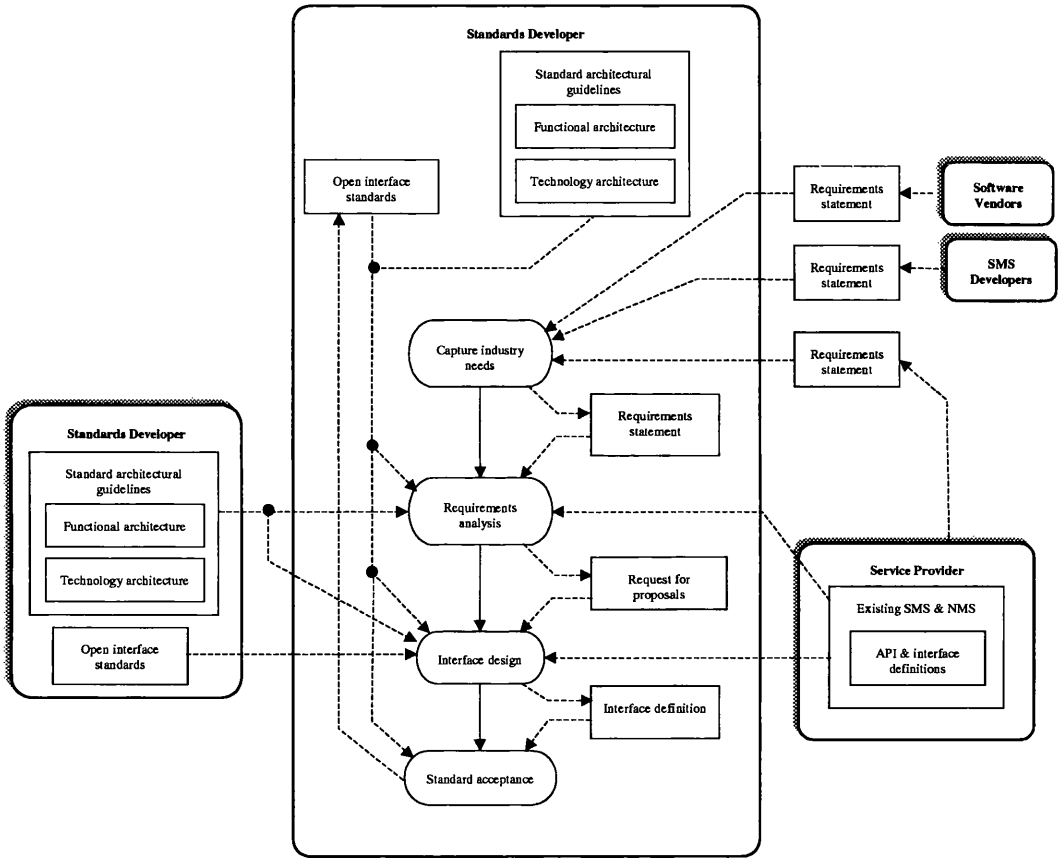


Figure 3-7: Process Model for Interface Standard Development

3.5.4 Generic Methodological Requirements

The above process models serve to identify the differences between the core development processes typically performed by the three SMS development stakeholder types. However, in order to define a common development framework that is practical for all three stakeholder types to use, the most promising approach

may be to focus on where the processes are most similar. Solutions to problems in these areas will have a higher likelihood of being widely understood by practitioners in organisations of all stakeholder types, thus enhancing the benefit of a common approach.

Organisational inertia makes imposing a common development framework across many different organisations very difficult. Arguments for established internal processes and architectures often outweigh any move to accept a common framework motivated solely by the need for smoother interactions with other organisations. This problem is especially acute when these organisations are of the same stakeholder type and are mutually perceived as competitors. A more productive approach to promoting a common development framework, therefore, may be to use the commonalties identified in the processes analyses, not to standardise those processes but to identify commonalties in the interactions between stakeholder types. The approach taken here to synthesising common methodological requirements is therefore to identify requirements that focus the interactions between stakeholders, but which accommodate the differences between the processes identified the stakeholder process models and those within real stakeholder instances.

The process model shown in Figure 3-8 depicts the commonalties between processes and interactions that exist between the different stakeholder process models. This is expressed in terms of processes with and interactions between a generalised SMS development stakeholder and a generalised collaborating stakeholder. The main differences between the stockholder's core processes are between standard definition and software implementation and between bespoke and generic software implementation, testing and deployment. However, the activities of requirements capture, requirement analysis and design are similar in all three models. In addition all three models require these activities to take into account architectural guidelines and existing solutions both from within the stakeholder and from other collaborating stakeholders. This common model forms the focus of the investigation of suitable methodological guidelines for a common development framework. It is used as a

template for examining the practicability of different methodological techniques is the case studies examined in the following chapter.

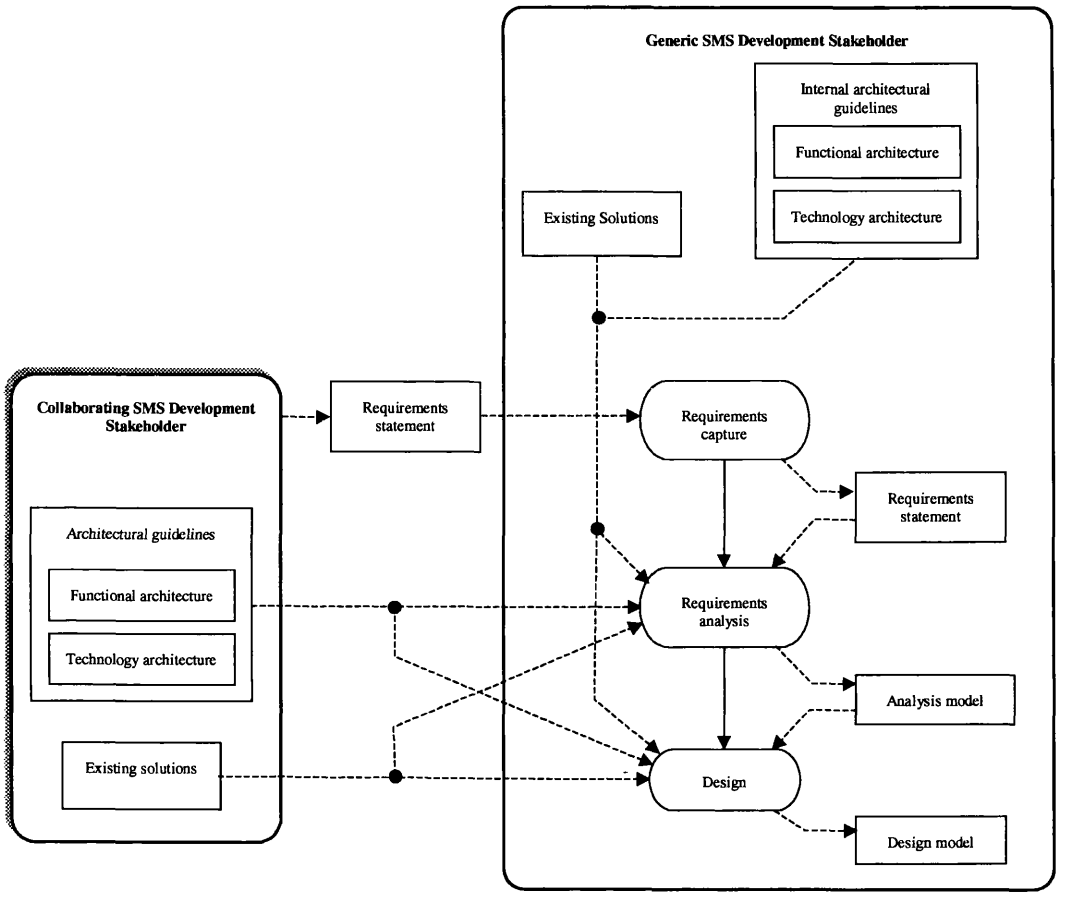


Figure 3-8: Generic SMS Development Stakeholder Process Model

4. Case Studies

This chapter describes five SMS development case studies, in which the author has been involved and has observed, evaluated and in some cases guided the development methodology used. These SMS development projects were undertaken as part of three different European Union (EU) funded collaborative research projects, which focussed on the development of management systems in an open services market. Each case study reflects a major management system development cycle in one of these projects. The case studies involve aspects of all the development phases of an SMS. The case studies were performed by collaborating teams of between one and two dozen, researchers, analysts and software engineers drawn from a range of companies, universities and research institutes across Europe. These case studies, therefore, do not reflect SMS development scenarios operating under real commercial pressures. They do represent, however, the opportunity to observe the development of SMS by teams who were committed to the use of standards and were also willing to both try out new development techniques and provide feedback on their experiences with them.

The first two case studies relate to the two development phases conducted in the *PREPARE* project. This project was funded under the EU RACE II programme and ran from January 1992 to December 1995. It aimed to investigate issues of applying TMN to the integration of service and network management in a multi-provider environment. In these case studies the author participated in a team that evaluated and refined the methodology used in the project. He was primarily responsible for publishing experiences and assessments of the methodology for the first phase in [lewis95a] and the second phase in [lewis95b] and [hall96].

The third and fourth case studies were conducted in the *Prospect* project. This was funded by the EU under the ACTS programme and ran from September 1995 to August 1998. It addressed the integration of service and network management with service control using a wider range of technologies than then specified in TMN,

principally CORBA. Both phases of Prospect followed an explicitly defined methodology defined by the author jointly with Vincent Wade of Trinity College Dublin and described in [wade97] and [wade98] for the first phase and [lewis99d] for the second phase. The author, however, was solely responsible for collecting and evaluating the experiences of the developers in forming the assessment of their usefulness presented in this chapter.

The final case study is from the *FlowThru* project. This was also funded under the EU's ACTS programme and specifically aimed to study methodological techniques for building management systems that satisfy business process requirements and which are constructed from reusable components. This project started in March 1998 and is due to complete in February 2000. The author played a primary role in defining the methodology and wider development framework used for this project, as published in [lewis99a] and [lewis99b]. He was also solely responsible for collecting and evaluating the developer's experiences from this case study.

Each case study is presented in a separate section. Each section: reviews the context within which the SMS development was undertaken; describes the development approach taken; summarises the mechanism to evaluate the approach and presents results of the evaluation. The description of the development approach for each case study is supplemented by examples of modelling notations and structures from the original working documents where appropriate

4.1 Case Study 1: OSI-SM and TMN

This case study is based on the management system development performed in the first phase of the PREPARE project. The material presented here is based on the authors contribution to the development experiences reported in [lewis95a] The PREPARE project was proposed with the aim of investigating network and service management issues in the multiple bearer and value added service provider context of a future deregulated European telecommunications market. The specific example selected for implementation in PREPARE phase 1 was of a value added service provider co-operating with multiple public bearer service providers to deliver a

Virtual Private Network (VPN) service to a distributed corporate customer. In order that these investigations had a realistic focus a broadband testbed network was constructed over which the VPN service would be demonstrated. This testbed consisted of several different but inter-working network technologies. Each of these sub-networks possessed its own network management system which were developed according to the principles laid down in the TMN recommendations and using platforms that implemented of the OSI CMIP mechanism. The investigation of such a multi-domain management involved the development of an architecture that allowed these separate network management systems to co-operate in providing end-to-end management services. This end-to-end architecture was also developed in accordance with TMN principles. The development approach taken was subject to the full rigour needed to implement a working system with the resulting management testbed being successfully demonstrated in public on 8th December 1994.

The make-up of the project consortium added a further importantly realistic aspect to the case study in that many of the roles played are relevant to the realisation of future multi-domain management services. The project partners and their relevant roles were:

- A public network operator (KTAS from Denmark), interested in integrating wide area network management with multi-domain service management.
- A public network equipment vendor (NKT from Denmark), interested in the management of Metropolitan Area Networks (MANs) and the management of heterogeneous network inter-working.
- A customer premises network and management platform vendor (IBM), who were interested in using their products (Token Ring and Netview/6000) in a multi-domain environment.
- A vendor of network management platforms (LMD Ericsson from Denmark in co-operation with Broadcom from Ireland), interested in the application of the TMOS Development Platform to value added service provision.

- Researchers into advanced network management techniques (University College London and GMD-FOKUS from Germany), interested in applying their platforms to the multi-domain environment.
- Researchers into multimedia applications (University College London), interested in the interactions of these applications with service and network management.

Project partners therefore brought to the project their own specific interests, which were often overlapping but sometimes different or even contradictory. Therefore, though not operating in a true commercial environment, the viewpoints of the Service Customer, the Service Provider (i.e. a value added and a bearer service provider), the Software Vendor (i.e. management platform vendors) were all genuinely represented. It can therefore be asserted that the methods chosen in arriving at this working TMN-based SMS implementation represent those that will have likely applicability to the SMS development stakeholders. In addition, though this was a prototype development exercise rather than a standards writing one, the lack of service management standards motivated the development of interfaces definitions that would be relatively generic and could therefore form a contribution to standardisation for inter-domain service management.

4.1.1 Development Approach

The process of defining management services and information models in an environment that contains several different types of player has received some theoretical attention at the time this work was conducted but the body of actual experience with large scale developments was still very limited. The main input that could be drawn upon at that time was ODP viewpoints, M.3020 and the TMF ensemble approach. For the first phase of PREPARE there were few examples of ODP viewpoints applied to management so this approach was regarded as too immature to apply here.

One limitation of the M.3020 and TMF ensemble approaches for PREPARE was their overall scope. The project required a methodology that was capable of

integrating the service specification, design and implementation phases of the TMN demonstrator. The scope of these methodologies only covered the requirements analysis and design processes as they were intended for defining interface standards rather than building systems to customer requirements. In addition, and of most significance for PREPARE, these two approaches are designed to support single interface design. Neither provided support for developing co-operative management systems with multiple interfaces.

The development team, though influenced by some of these approaches, did not follow any closely but synthesised its own methodology. A pragmatic approach was taken that was primarily driven by the experience in management system development and knowledge of contemporary development methodologies possessed by the team members. The approach was heavily influenced by the division between intra-domain and inter-domain management system development. This reflected the make up of the project testbed that had an ATM Wide Area Network (WAN), an ATM Customer Premises Network (CPN), a DQDB Metropolitan Area Network (MAN) and a Token Ring Local Area Network (LAN) all being provided by different partners. The partners responsible for providing each network also provided the accompanying implementations of EML, NML and SML OSFs. Each network type therefore possessed its own TMN up to and including the SML, with each TMN being modelled as existing in a separate organisational domain. The methodological approach therefore focussed on the development of the inter-domain interfaces since this was where partners would collaborate and would therefore gain most benefit from a common development approach. No attempt was made to prescribe how individual intra-domain systems were developed.

Against this background the work proceeded as four separate, but inter-linked activities:

- *Scenario Definition:* This activity produced a set of scenarios that detailed what would be demonstrated over the management testbed. Due to the large number of participants, components and requirements involved, these scenarios were seen

as essential in order to focus the work onto a manageable subset of demonstrable operations while at the same time presenting a coherent description of what was to be demonstrated.

- *TMN Architecture Definition:* This activity had to interpret the TMN recommendations in order to produce a functional architecture that specified how the OSFs in the different domains should be connected to each other via reference points in order to provide end-to-end services.
- *Management Service Definition:* This activity defined a set of services that operated between the different management OSFs in accordance to the Abstract Service Definition Convention (ASDC) Recommendation [x407].
- *Management Information Modelling:* This activity defined the information models required by the various OSFs interfaces that were involved in inter-domain relationships, using GDMO.

Due to restrictions of time and man-power these groups contained only a small core of overlapping personnel and were generally conducted in parallel. It was intended that the Management Service Definition and the Management Information Modelling be focussed on satisfying the requirements laid out by the Scenario Definition and on defining the inter-domain reference points identified in the TMN Architecture Definition.

At the beginning of 1993 a review was conducted of the work performed in the first stages of design and its suitability for supporting the subsequent implementation work. The output from the Scenario Definition activity had described the roles of the human users and organisations involved in the VPN service as well as the motivations for the operations they performed. This was supplemented by descriptions of the commercial service that the VPN provider should provide to its customers in terms of contractual responsibilities. The TMN Architecture Definition group had identified the OSF required to provide end-to-end VPN services and the different reference points required between them. The architecture was structured according to the management functions types, reference point types and logical

layering of [m3010]. The architecture was designed based on the following principles:

- Each organisational stakeholder, including customers, had its own TMN.
- Organisations which own physical networks have within their TMN an OSF specific to the particular network technology being managed by that TMN, i.e. a Network OSF (N_OSF).
- Each TMN has a Service OSF (S_OSF) implementing service management functions associated with that particular domain, and which takes part in providing the distributed end-to-end service management services.
- The VPN provider, which did not operate a network, has a TMN continuing only an S_OSF.

As a result N_OSFs inter-operate with the S_OSFs in their own TMN (via q-type reference points) and S_OSFs inter-operate with S_OSFs in other TMNs (via x-type reference). Figure 4-1 presents an overview of the resulting functional management architecture.

During the architecture definition activity, attention was paid to explicitly addressing the non-functional requirements imposed by the scope of partners' interests and the platforms available to partners, as well as reducing the overall complexity of the information modelling tasks by minimising the number of inter-domain reference points involved. By addressing these issues at the architectural stage of the design process it was found subsequently easier to split the work between relatively independent groups addressing different areas of the functional architecture.

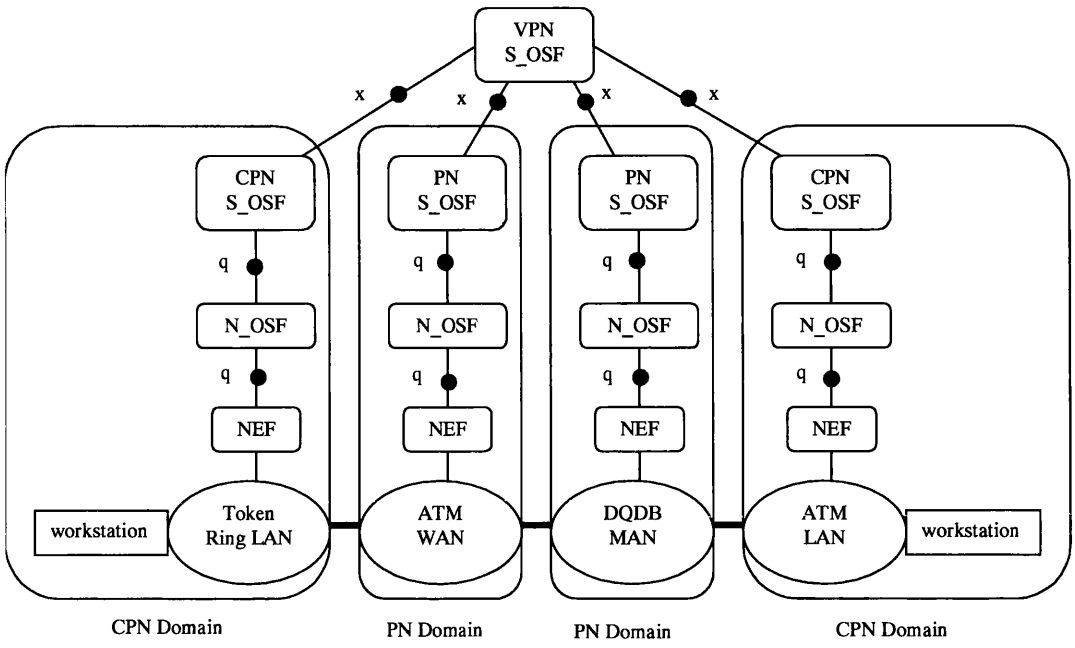


Figure 4-1: TMN Functional Architecture for PREPARE Phase 1

It was apparent from the review that the scenarios contributed greatly to the teams collective understanding of the problem while the architecture was generally agreed upon as being suitable for the implementation of the VPN service. However it was also recognised that the outputs from the management services and information modelling groups suffered in many respects. Firstly, these two sets of output were not mutually consistent nor were they totally aligned with the output of the scenarios and architecture groups. Co-ordinating this work while running the groups in parallel had apparently proved too complex a task given the human resources available. Secondly it was felt that, given the goal of demonstrating the scenarios, the management service and information model definitions were not complete and did not contain the level of detail required by the implementers.

Though a combination of the GDMO and ASDC descriptions of an interface were expected to provide a clear definition of the reference points, there was no formal mappings between GDMO and ASDC and no automated support for maintaining such mappings as the two specification were developed in parallel. This made this

approach problematic. In addition the behaviour description of individual MOs and ASDC operations were not sufficient in defining the behaviour of the OSF as a whole. The development approach was therefore modified by abandoning the further definition of management services and concentrating on refining the scenario description. The existing scenarios were refined from a level where they described enterprise roles and their relationships, to one where the same scenarios were described in terms of OSFs with detailed definitions of the management information flowing between them given as sequence diagrams. By adopting this technique, a full GDMO specification for the inter-domain reference points was quickly arrived at. This approach also had the intrinsic advantages of ensuring that all information modelling was directly focused on the desired implementation areas and, through scenario descriptions, providing an informal but relatively brief description of the behaviour of the collaborating OSFs.

The entire information model for all inter-domain interfaces was maintained in a single document referred to as the Implementer's Hand Book (IHB). It was apparent that although the aim at this stage of the design work was to arrive at a stable version of the information model, there would inevitably be changes required to the IHB as the understanding of the problem grew. For this reason the IHB was maintained as a living document. This task was made considerably easier with the help of Damocles a GDMO parsing and checking tool developed by GMD-FOKUS. This was used to check the IHB for GDMO syntax errors and open references, but more importantly it assisted in the manual checking for consistency and completeness throughout the information model. This was especially useful considering the number of partners involved in contributing to this document. A mechanism for requesting updates or modifications to the information model was also adopted since changes inevitably effected more than one partner's implementation work.

As the IHB became stable and the inter-domain implementation began, the planning for integrating the various hardware and software components commenced. This was conducted broadly following the IEEE standard 829-1983 [ieee829] which involved the generation of Test Design Specifications (TDSs) for all tests that would involve

components from one or more partners. When this was performed for inter-domain management software components some interesting effects were observed. Firstly, the refined scenario descriptions proved to be ideal templates for defining the interactions that should be tested, ensuring once again that the work performed directly supported the final aims of the project. Secondly, the TDSs were written to a level of detail that defined the actual CMIS primitives that should be exchanged between the OSFs and the information content required. This process of writing the TDS to such a level of detail provided much invaluable insight for the implementers in that it raised many issues that had not yet been recognised and allowed these problems to be resolved before the implementation work had progressed too far. These problems often related to the relationships between different MO classes which supported different OSF management functions, but which were both related to the same underlying resource. In addition, problems related to differences between the structure of information at inter-domain reference points and at the separately developed intra-domain reference points revealed themselves at this stage. This indicated that the level of detail used at the testing stage should ideally have been addressed at the design stage.

4.1.2 Evaluation and Results

The evaluation approach taken was purely anecdotal, based on the author's own experiences and those elicited during discussions with other PREPARE team members. The overall development process taken in this case study is depicted in Figure 4-2.

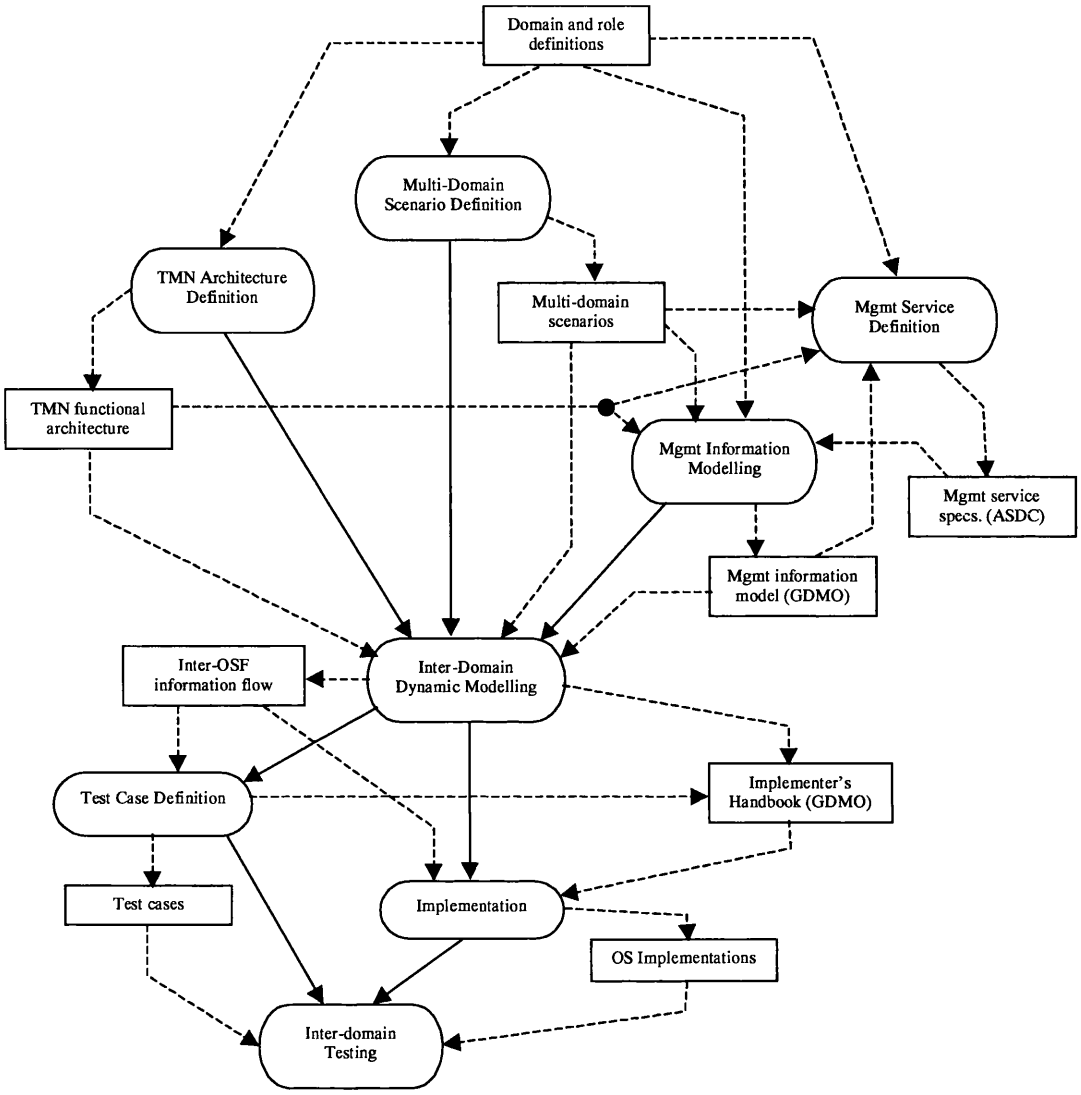


Figure 4-2: Development Process for Case Study 1

With respect to the SMS development stakeholder model, this case study represents a green-field situation where there are no existing service management standards or software products to draw upon and the SMS Developers for different Service Providers must collaborate to agree inter-domain interfaces on a case by case basis. However, as the intention was to generate generic reference point definitions, some of the experiences of this case study could be relevant to the development processes within the Standard Developer stakeholder.

This case study provide some evidence to support the statement in the thesis hypothesis that development techniques that already exist for network management system development are inadequate for SMS development. Network management system development techniques are typically extended from the information modelling paradigm used for network element modelling, where the primary modelling activity is the object-oriented modelling of the physical and logical resources to be managed. When applied to OSI-SM, the resulting managed objects define the functionality of an agent entity that may be accessed by a manager, i.e. it is focused on the definition of a single manager-agent interface. This approach is insufficient for an SMS which, as exemplified in this case study, must typically operate in an environment of multiple, interoperating functional units that play both manager and agent roles. Here functional units will exhibit multiple collaborative relationships, rather than the strict hierarchical relationships typical of network management. Approaches to defining a single interface, such as M.3020 or the TMF ensemble approach, are therefore, insufficient for analysing the behaviour of a service management OSF playing multiple roles and for designing the multiple, interrelated agent interfaces that implement these roles.

The major methodological problem encountered in this case study was in attempting the modelling of both the management information and management functions visible at a reference point in parallel. This problem was exacerbated by the separate notations used for these models, i.e. GDMO and ASDC. These models were closely linked, with the structure of information being influenced by the functions required to be performed with it while the choice of functions was influenced by the information available. The development of these models should have been much more closely integrated but this was impeded by the lack of mappings between these notations.

In M.3020, management services and management functions are both regarded as reusable entities. However, abandoning the specification of management functions in this case study precluded their availability for later reuse, possibly at another reference point of the same OSF. Reuse was therefore limited in this case study to

the use of standard ASN.1 types or standard abstract MO class definitions during information modelling. Reuse of code or binaries was also ruled out in the implementation stage by the presence of different, proprietary API's in the various CMIS platforms used in the project.

Of the modelling techniques that were applied in this case study, the most useful was found to be the application of scenarios. These aided greatly in the communal understanding of the multi-domain problems being addressed and in co-ordinating the individual modelling efforts required for a multi-OSF interaction scenarios that were required. Finally, it was also found that scenario-based development proved useful for generating test cases, where test cases were based on the initial scenarios, but instantiated with specific preconditions and operational parameters.

4.2 Case Study 2: Responsibility and Computational Modelling

This case study is based on development performed in the second phase of the PREPARE project. The development experiences reported here are based on work by the authors published in [lewis95b] and [hall96]. The second phase of the PREPARE built upon the first phase in terms of the methodological experience gained, the construction of the management testbed and the user services that operated over it [lewis94]. The second phase differed from the first in that the enterprise situation was more complicated, involving more service providers and more relationships between service providers, with the range of scenarios being addressed being more ambitious. There was also a change in architectural emphasis from simply producing service level OSFs in each domain, to adding WSFs with rich functionality for service and network administrators. It was however similar to the first phase in that the management architecture had a TMN-based structure.

The enterprise situation modelled a Multimedia Conferencing (MMC) teleservice provider and a Multimedia Mail Global Store (GS) teleservice provider which provided their services to users on CPNs. The teleservice providers used the services of a separate VPN provider to manage end-to-end network resources over multiple

public network domains and the CPN domains in support of the teleservice providers' communication needs. This management testbed was successfully integrated and tested and then demonstrated publicly on 30th November 1995.

4.2.1 Development Approach

Based largely on the experiences of the first phase, it was felt that a more cohesive development approach was required. The requirements capture, analysis, design, implementation and testing was therefore performed under one group which would split into subgroups at various stages to address clearly defined functional areas rather than splitting into groups addressing the different development activities. The development approach taken can be broken down into the following activities:

- *Enterprise Modelling and Scenario Description:* This described the organisational context in which the management systems were required to operate by identifying the organisational domains and human operator roles and describing their interactions as a set of scenario descriptions.
- *Role Specifications:* These provided a way of describing in more detail the requirements of the involved organisations through the definition of responsibilities for individual roles identified in the enterprise model and a way of mapping these requirements to lower level management function requirements.
- *TMN Architecture Definition:* As in the first phase of PREPARE, this defined the functional architecture of the TMN systems that would provide the framework for the more detailed design work. This was expressing in terms of NEF, OSFs and WSFs, their positioning within logical layers and organisational domains and the identification of reference points required between.
- *Information Modelling and Information Flow Analysis:* This involved the identification of information required by the management functions identified in the scenarios and role specification. The analysis is performed in terms of information models defined for each domain and inter-OSF sequence diagrams

showing information flow over the reference points defined in the TMN functional architecture.

- Design of Functional Units: This involved the functional decomposition and design of the various OSFs, WSFs and NEFs. The development of the latter is not discussed further here.

These activities were not addressed is a strict sequence, but to an extent were interleaved with some being revisited after the initial work on others had provided clearer insight into the requirement upon them. Each of these activities is now described in more detail.

4.2.1.1 Enterprise Modelling and Scenarios

This activity identified the organisational stakeholders and their characteristics (e.g. core business areas), with the focus on the objectives for their involvement in the scenarios in order to identify the high-level requirements on the system. Scenarios concentrated on inter-domain aspects, i.e. inter-organisational relationships where agents of the organisations interacting on behalf of their organisations in specific roles. To help identify these human roles in a consistent manner, the organisations were classified by a set of abstract business roles. These were based on a separation between a service provision relationship and the accompanying commercial relationship. These relationships are described in terms of business related meta-roles. For the service provision relationship a service *supplier* provides a service to the service *user*. For the commercial relationship a service *vendor* provides a service to a service *customer*. The following abstract organisational business roles were therefore defined in terms of the above meta-roles:

- A *service consumer* is an organisation acting as both service user and customer with another single organisation acting as corresponding service supplier and vendor.
- A *service provider* is an organisation acting as both service supplier and vendor with another single organisation acting as user and customer.

- An *indirect service consumer* is an organisation acting as service user and customer but where the corresponding supplier and vendor are separate organisations.
- An *indirect service provider* is an organisation acting as service supplier and vendor but where the user and customer are separate organisations.

4.2.1.2 Role Specifications

This activity aimed to further describe the relationships between these roles that place requirements on inter-domain management functionality. Role specifications were adopted as a means of ensuring that the management functionality required by the role holders in the scenarios was adequately described and provided full requirements for the inter-domain reference points.

A common role specification template was adopted in order to structure the description of what the role requires, and to facilitate refinement of the role specification down to the operations on the managed resources that would eventually be modelled as MOs at a reference point. This template was based on the work of the ESPRIT project ORDIT, which investigated the organisational requirements for information technology systems by examining roles and responsibilities within an organisation [strens][dobson]. In PREPARE, the ORDIT concepts were adapted for the specific needs of the role specification work and inter-domain service management. The role specification template therefore included for each role holder the responsibilities of the role holder in relation to other role holders. The responsibilities were then refined into finer grained obligations that needed to be discharged by the role holder in order to meet the responsibilities of the role. Obligations were then decomposed into activities that needed to be carried out to enable the role holder to fulfil the obligations deriving from the responsibilities and the resources and access rights required to enable the role holder to carry out an activity. The specific human role holders required for the case study were identified via the scenario descriptions, and were derived from the abstract business roles. The identification of role holders was split between those dealing with the contractual

and financial aspects of service management and those dealing with the more technical and operational aspects, thus reflecting the similar separation in the business meta-roles.

VPN Service Manager

Responsibility #1) Responsibility (to VPN End user Agent) to ensure end-to-end communication paths are set up to satisfy their communication requirements.

Obligation #1) To ensure end-to-end communication paths are set up between end points associated with the VPN end users with the QoS requested by the VPN End user Agent.

Activity #1) Request a user stream from the VPN Service Administrator specifying the end points and the QoS parameters
Resource #2) User stream (create)
Activity #2) Modify user streams as required by the VPN End user Agent.
Resource #1) Termination point (read)
Resource #2) User stream (read, update, delete)

VPN Service Administrator

Responsibility #1) Responsibility (to the VPN Service Manager) to ensure that the sufficient resources have been allocated in the VPN.

Obligation #1) To reserve requested resources in the public network operator domain

Activity #1) Request that a network link is reserved over the public network operator domain.
Resource #1) Network Link to VPLine mapping and representation (create, read, modify, delete)
Resource #2) Translation point (read)
Obligation #2) To reserve resources in the private network domain

Activity #2) Request or verify that a network link is reserved over the private network operator domain.
Resource #3) Network Link (create, read, modify, delete)
Resource #4) Termination point (read)
Responsibility #3) Responsibility (to the VPN Service Manager) for the end to end communication stream provision and maintenance

Obligation #3) Receive, verify and acknowledge the request for a user steam

Activity #3) Verify available connectivity reservation
Resource #3) Network link (read)
Activity #4) Allocate reserved capacity in public network operator domain
Resource #2) Translation point (read)
Resource #5) User stream (create)
Activity #5) Allocate reserved capacity in private network operator domains
Resource #4) Termination point (read)
Resource #5) User stream (create)
Obligation #4) Report the request for a user stream creation to customer service administrators.

Activity #6) When subscribed to send notifications on changes in the VPN
Resource #6) User stream creation creation notification (create)

Figure 4-3: Example of a Role Specification for a VPN Service Manager Role

Organisations that acted as a service consumer had a *financial agent* role holder broadly responsible for locating new services, subscribing to them, paying the bills

and terminating subscriptions. Organisations that acted as service vendors had a financial agent role responsible for receiving requests for service subscription, granting or denying the request, sending bills and terminating the service. Organisations that acted as service users had a *service manager* role that dealt with the operational aspects of service usage while organisations that acted as service suppliers have a *service administrator* role that dealt with the operational side of service provision. Each organisation also has an owner role to which the organisation's other roles are ultimately responsible and which was included to ensure the completeness of the role specification set. An example of role definition for the VPN provider taken from [hall96] is given in Figure 4-3.

4.2.1.3 TMN Architecture Definition

This activity followed the same principles is in Case Study 1. The resulting functional architecture is depicted in Figure 4-4.

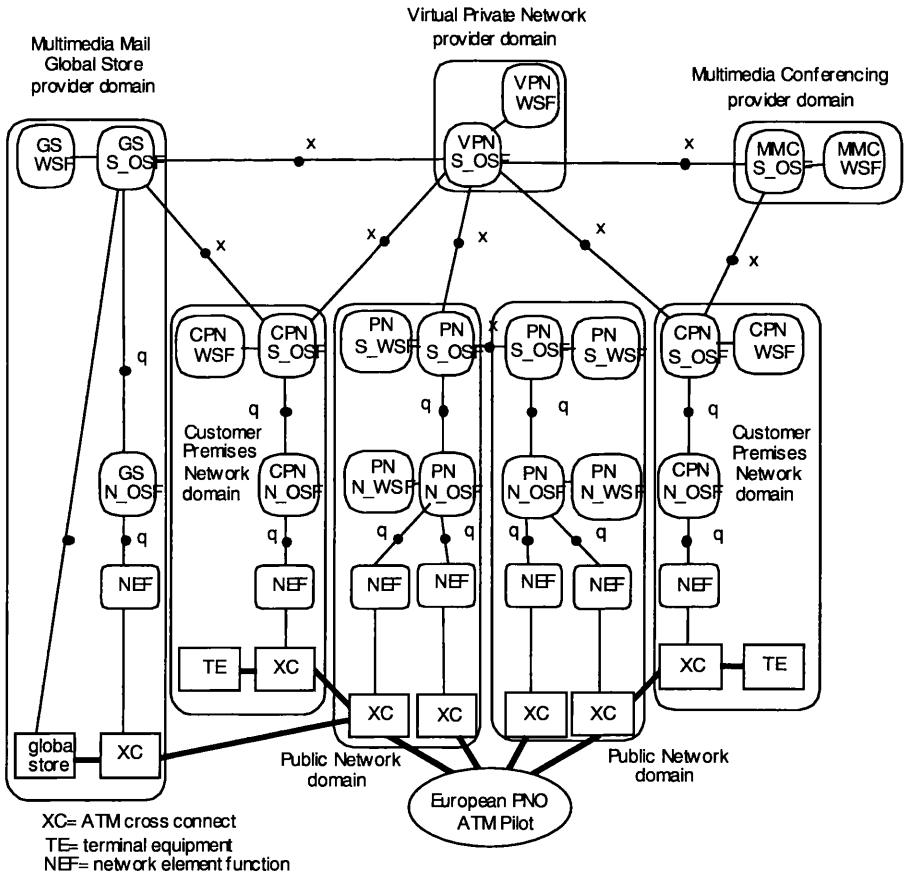


Figure 4-4: TMN Functional Architecture for PREPARE Phase 2

4.2.1.4 Information Models and Information Flows

Initial information models based on the requirements imposed by the enterprise model and the scenario descriptions were made more concrete by the identification of resources in the role specification. These initial MO descriptions were expressed simply as text description of what the MO's represented. As resources from the role specifications were associated though role to organisational domains, these MOs were straight-forwardly associated with the S_OSFs identified for each domain in the TMN functional architecture. Inspired by its usage in the ODP information model as applied in TINA, the initial information models were enhanced with OMT class diagrams showing the relationships between the MOs supported by an individual OSF, and in some cases the relationships to MO's in other OSF. An example for the VPN S_OSF is given in Figure 4-5, with relationships to the MO's in the ATM N_OSF shown in grey.

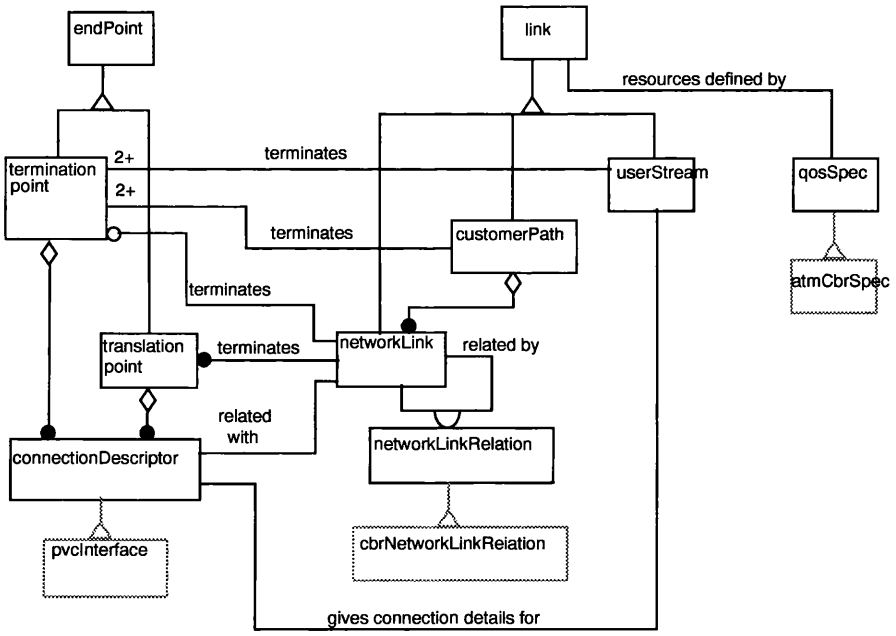


Figure 4-5: VPN Information Model

Inter-OSF information flows were then generated and refined, detailing how the management activities outlined in the scenarios were accomplished by operations on managed objects. Information flows were described in terms of CMIS message flows between OSFs. Thus they identified the MOs present at a reference point, which operations were performed on them, and with which attributes and values. An example information flow taken from [hall96] is shown in Figure 4-6. The information models and flows were designed in an iterative fashion, since information flows identified missing information that needed to be included in the information model specification and subsequently verified through updated information flows.

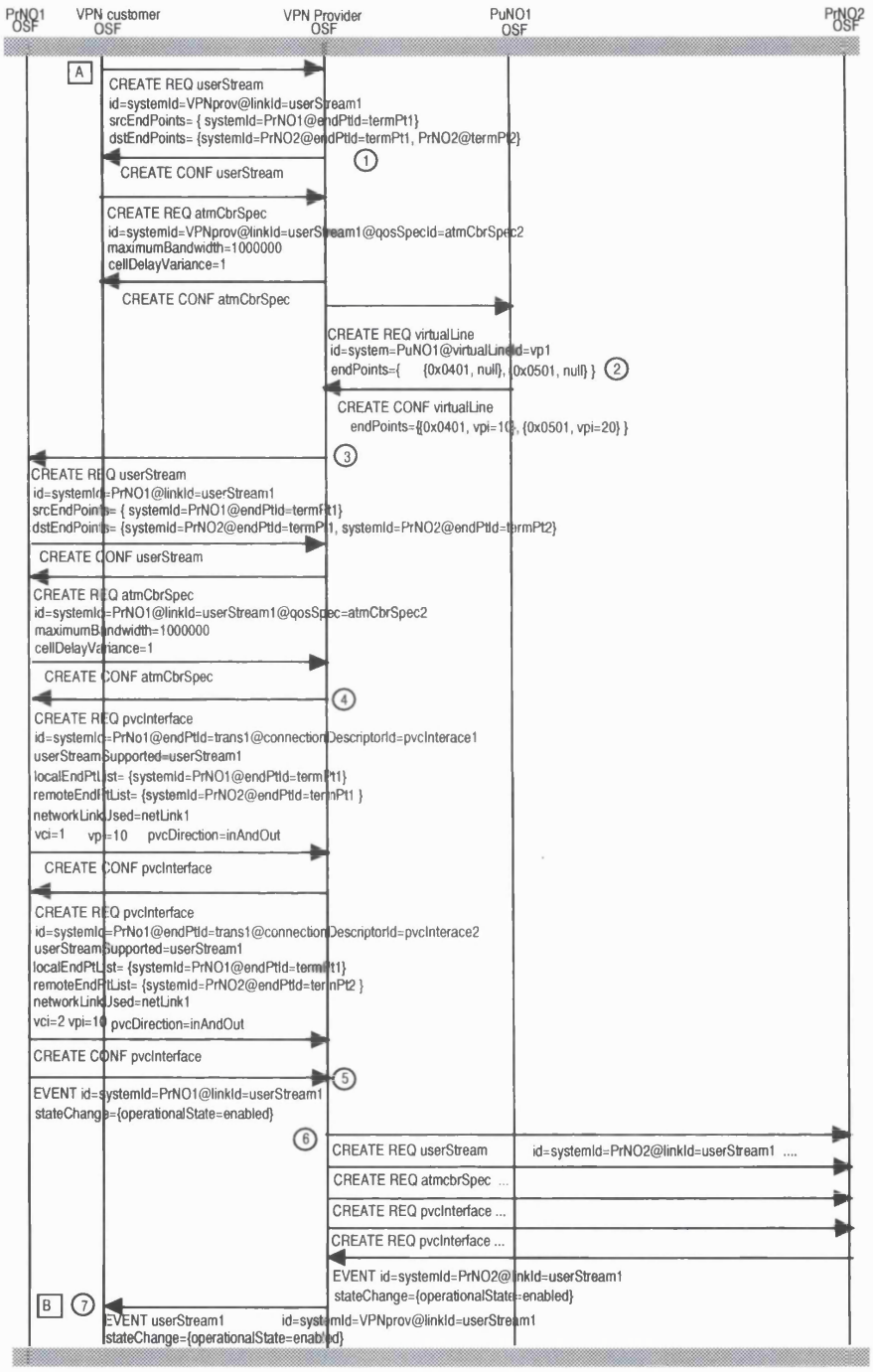


Figure 4-6: Example of Information Flow Sequence Diagram for the Creation of a VPN User Stream

4.2.1.5 Management Function Design

The functionality of the OSFs identified in the TMN architecture was generally governed by the requirements of the scenarios and role specification and the resulting functional interactions across reference points defined in terms of the information models and the information flows. Once this level of detail had been achieved the functional design of individual OSFs was left largely to the judgement of individual designers. In a few cases, however, where OSFs developed by different partners shared common functional requirements, a more fine-grained approach was taken to the functional decomposition of the OSFs. This work took an object oriented approach loosely based on the ODP computation viewpoint as applied by the TINA-C. This involved defining functional building blocks that addressed specific functional areas, e.g. billing or customer interface functions, and that could be used in different OSFs. These building blocks were then further decomposed into computational objects (COs) that provided both the functional structure of the building blocks and the interfaces offered by this functional building block to other functional building blocks in the same of separate OSFs. The COs were defined with multiple interfaces to explicitly differentiate between the functions and access rights required by the different roles played by OSFs as identified in the role specifications. The final design then consisted of mapping these COs onto engineering objects that implemented the OSF functionality. Where this was performed, it was done so in a proprietary manner as described in [tiropanis97], i.e. proprietary APIs were defined for CO implementation interfaces. Within a TMN platform, inter-OS communication is performed by creation, deletion or attribute change operations on MOs implementing an OS's interface. All internal CO communication, however, potentially all could be via invocations on the proprietary CO interface API. However, in order to efficiently integrate inter-OS and intra-OS interactions, most communication between COs was performed by one CO operating on MOs and others receiving notification of this using the same mechanism used to generate inter-OS CMIP notifications. In other words the COs communicated via an MO-based notification mechanism using the existing internal CMIS event forwarding

discriminator mechanism. Hence the propriety inter-CO API was implemented through operations on existing MOs and only through direct invocations or CO interfaces when no suitable MO definitions existed in the reference point definition. The definition of the COs used an augmented version of TINA's ODL. An example of the textual part of this notation for a single COs is shown in Figure 4-7.

```
COMPUTATIONAL_OBJECT_CLASS      e2eResourceAllocationMgr
SERVER_INTERFACES
        NAME                    csmControlInterface
CLIENT_INTERFACES
        NAME                    statusInterface
BEHAVIOUR

END_TEMPLATE

COMPUTAIOTNAL_INTERFACE csmControlInterface

        OPERATION       createUserStream

        OPERATION       deleteUserStream

        OPERATION       modifyQos

        OPERATION       addSourceEndPoint

        OPERATION       removeSourceEndPoint

        OPERATION       addDestinationEndPoint

        OPERATION       disableuserStream

        OPERATION       enableUserStream

BEHAVIOUR

END TEMPLATE::

OPERATION createUserStream

        INPUT PARAMETERS

                sourceEndPoints:        SET OF endPoints

                destinationEndPoints: SET OF {SET OF endPoints}

                qualityOfService:       SET OF REAL

        OUTPUT PARAMETERS

                userStreamId:                   OBJECT IDENTIFIER

        RAISED EXEPTIONS

BEHAVIOUR

END_TEMPLATE
```

Figure 4-7: Example of CO Textual ODL Definition

An example of the graphical notation of ODL showing the decomposition of the VPN S_OSF and WSF into COs grouped as building blocks is given in Figure 4-8.

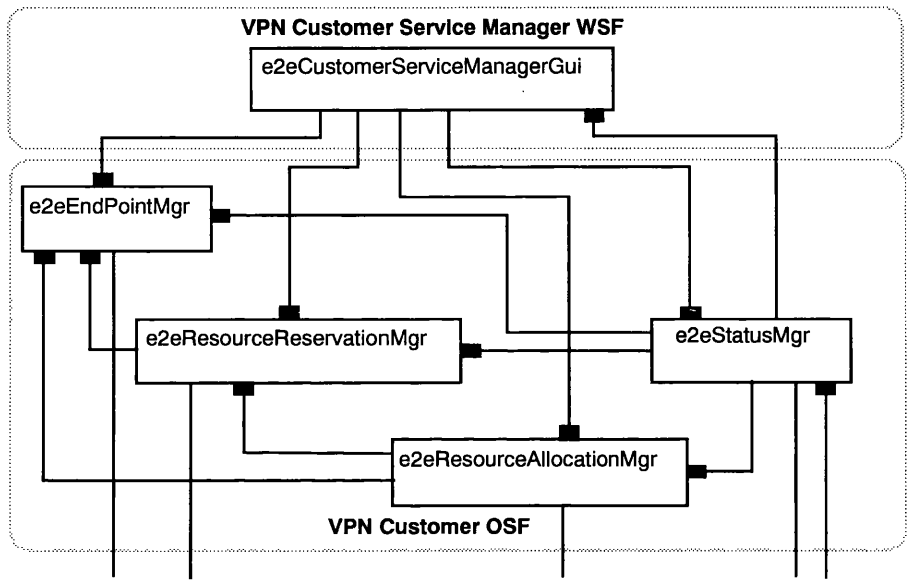


Figure 4-8: Example of ODL Diagram Showing COs in an OSF and a WSF

In cases where an organisation played more than one business role, its service layer OSF was decomposed into OSFs performing individual roles, e.g. the multimedia conference provider domain contained S_OSFs for both the VPN customer role functions and MMC provider role functions. This allowed OSFs to become units of reuse, e.g. the VPN customer OSF was instantiated in several organisational domains. From the computational viewpoint, such a reusable OSF was represented as a single building block. As roles had been defined along lines reflecting functional divisions in service management, e.g. separating out roles for service provision, accounting management and resource/network management, then the OSFs also reflected this natural split, which assisted in their reuse.

In mapping several OSF reference points into a single OS interface, the useful split between the different manager role related functions offered by the OSFs was lost, i.e. the different functions of separate OSF roles were not visible in the corresponding OS's agent interface. This was addressed in the project by defining

MOs that would form the head of the naming tree of MO which reflected the abstract business roles played by the constituent OSFs at that interface. Instantiating such MIB sub-tree for each business relationship that fulfilled an abstract business role provided a mechanism for naming and locating required portions of a service management interface, ensuring that role separations had operational significance. As the testbed included platforms that supported combined X.500/X.700 global distinguished names could be used for MOs, e.g. (cs=uk, o=ucl, ou=cs, system=vpn-os, indirectProviderSvcInstance=cust1, userStream=us1)

WSFs provided the representation of systems and sub-systems as relevant and needed by a role holder, taking various concerns into account. The WSF's design depended to a large extent on platform technologies, in that such platforms often have individual style guides prescribing many aspects of the GUI, for instance use of colours and maps, window layout and menus. The role specifications, however, provided important indications of what was to be represented on the screen (the resources the role holder managed), and the capabilities over these resources which are available to the role holder, which for instance provided indications of the contents of menus associated with each resource.

4.2.2 Evaluation and Results

The effectiveness and usefulness of the various methodological and architectural techniques use in this case study were assessed both through the author's own experiences in leading the working group that performed the analysis and design of the system, and through informal discussion with and feedback from the developers.

This case study represents a small increment on the first, dealing as it does with a refinement of the same multi-domain, TMN-based system. However it introduces two new development techniques were introduced: the modelling of responsibilities during requirements analysis and the use of computational modelling in the design activity. These are summarised in Figure 4-9, which highlights the difference to the process used in the first phase of PREPARE.

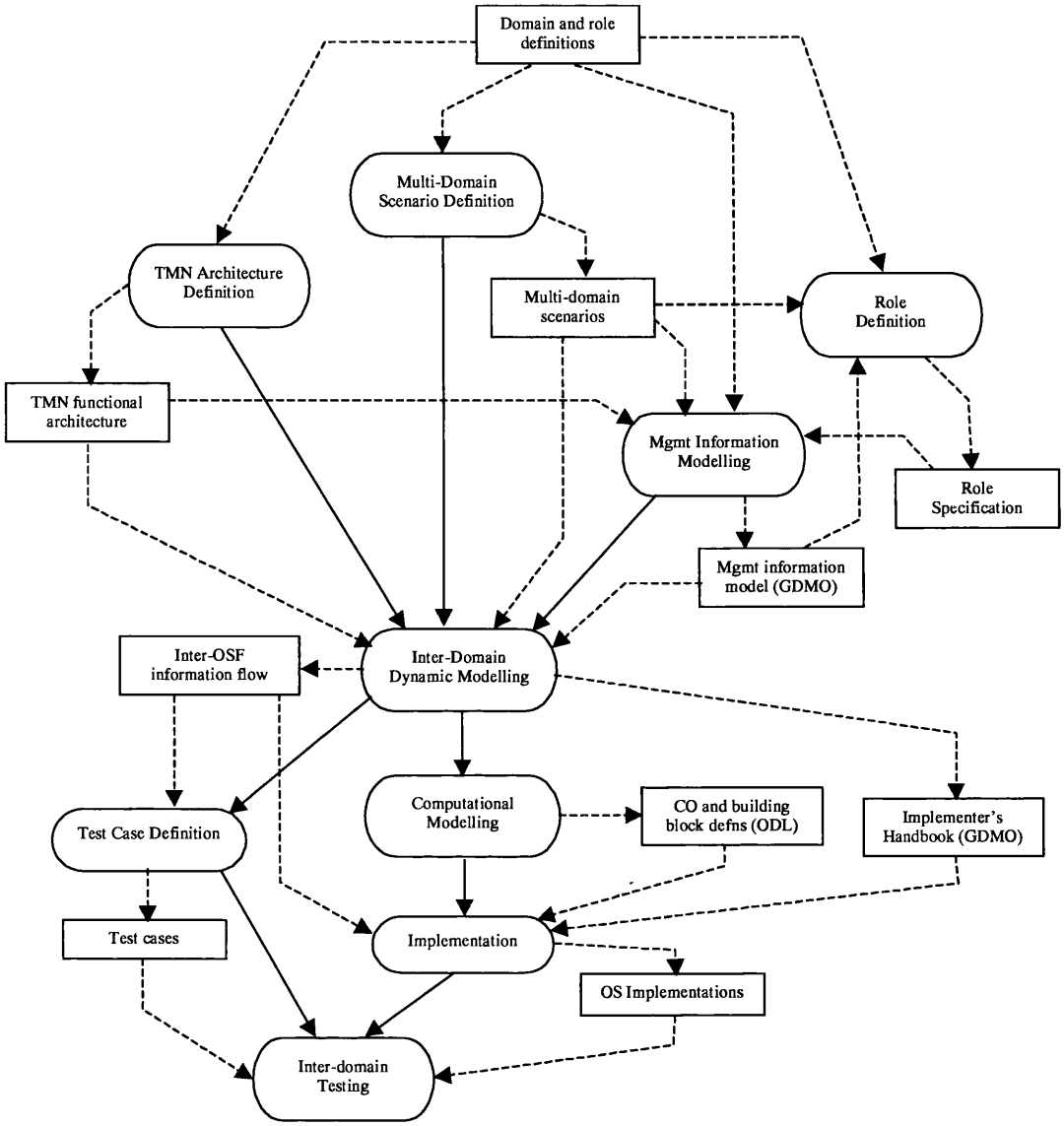


Figure 4-9: Development Process for Case Study 2

As with Case Study 1, the M.3020 and the TMF ensemble approach were not found to provide sufficient guidance. Both approaches assumed that only a single manager-agent interface was being addressed and thus were able to make information modelling subservient to the functional decomposition, forming only the last part of the methodology. This makes these approaches vulnerable to the type of problem observed with other methodologies driven by functional decomposition, e.g. diffusion of control of data. Therefore, some of the potential benefits of object-

orientation gained from the use of GDMO, such as information hiding, are not necessarily present in the design of a MIB and therefore not of potential benefit to its implementation. Where the problem at hand involved manageable resources being represented on both sides of an interface, as was the case in the design of the PREPARE VPN to CPN interface, information modelling must be promoted to an earlier stage in the analysis and design process than offered by M.3020. This allows the functional decomposition of both manager-agent interfaces to be informed by the informational composition of the problem, and also aids in ensuring the consistent management of information between the two entities communication over the interface.

The use of responsibility modelling was found to compliment rather than negate the usefulness of scenarios. Scenarios described the sequence of events that may occur over time between a set of organisations and management users. In practice this technique was used in the case study largely to help clarify the complex situations where there were interactions between two or more organisations were involved. Responsibility modelling identified the responsibilities a role in one organisation had with respect to a role in another. Responsibility modelling therefore focused on the set of one-to-one relationships between organisations, thus not revealing the multi-domain interaction view given by scenario modelling. However, role specifications tended to lead to a more comprehensive set of requirements on the individual OSFs than was obtained from the multi-domain scenarios.

The mechanism for refining these respective models led to a consistent view of the design of the various OSFs and their inter-domain reference points. Scenario modelling was refined by applying the scenarios to the functional architecture overlaid on the organisational structure, and thus revealed inter-domain OSF interactions. As observed in Case Study 1, this was helpful in modelling information that had to span more than one domain or that had to be exchanged between domains. The scenario-based interactions were refined down to the level of inter-OSF sequence diagrams showing the CMIS operation needed to perform a scenario or portion of a scenario. This was regarded by developers as a key design tool in

evolving the OSF reference point definitions into GDMO specifications, in contrast to its application in Case Study 1 which was restricted to test case development.

The refinement of responsibilities into obligations and then into activities and resources using the ORDIT technique was not so well received by developers. This tended to be performed in a bottom-up manner once some indication of the resources in each domain had been formed during information modelling. However, it did prove useful in providing a consistency check between the functionality required from a domain's S_OSF in order to satisfy its contractual responsibilities and both the information held by that domain and the operations that were permitted on that information, i.e. create, read, delete, modify. This could therefore identify some information and operations that a domain must offer at an interface that may not have been identified by refinement of the more narrowly focussed scenario descriptions.

The computational modelling introduced in this case study addressed the decomposition of OSFs into functional units that were both more manageable and potentially reusable. TMN allows OSs to be composed of multiple OSF in order to achieve a more fine-grained functional decomposition, as demonstrated in [griffin96]. However the interfaces to such OSs still have to be expressed in terms of MO classes and manager-agent operations. Whether OSFs can be implemented efficiently and flexibly as a reusable functional unit within an OS depends on the structure of the TMN platform used and is not addressed by the TMN standards.

The functional decomposition approach used drew heavily from TINA concepts of computational modelling where the unit of functional decomposition was the computational object. Though the ODL notation was helpful in defining the different interfaces that objects offered each other, these interface definitions did not map well to their physical implementation within an OS. The solution adopted, i.e., using MOs to propagate notifications between COs, offered the advantage of being very flexible. New functionality could be added by introducing a new CO that simply listened to MO operations made by existing COs that need not be aware of the new CO. This

represented a high degree of decoupling between COs which could potentially be exploited in their reuse elsewhere. By convention individual computation objects were given responsibility for specific MO classes. However this relationship was not directly supported by the management platform or by the ODL notation so therefore it was not possible to ensure that only certain MOs were accessed by a CO. The reusability of a building block of COs was therefore linked to the presence of specific MO class implementations, which is not expressed by the ODL definitions, thus making reuse more problematic.

Finally, it should be noted that though notations from TINA's application of the ODP viewpoints were used in this case study, the ODP concept of consistent, orthogonal viewpoint were not explicitly applied. The next case study provides an example of such a development process.

4.3 Case Study 3: ODP Viewpoints

This case study is based on multi-domain SMS development that occurred as part of the first development phase of the Prospect project. The methodological approach has already been reported by the author in collaboration with others in [wade97] and has been disseminated by the EU's ACTS programme as a guidelines recommending best practice to industry [wade98]. This case study presents the author's own contribution to this work. The example used for this case study is the development of a multi-domain subscription management service for a Tele-Educational Service (TES) provider. The tele-education service being managed is composed of several Multi-Media Tele-Services (MMTS) provided by separate service provider, i.e. two different WWW-based information services, a multimedia conferencing service and a VPN service and ATM service. The systems developed were successfully trialed by multiple users across a pan-European ATM network in March 1997.

4.3.1 Development Approach

The Prospect consortium mostly consisted of members from the PREPARE project so the development approach was able to draw upon the experiences described in the

previous two case studies. However, the SMSs constructed for this project were implemented using CORBA platforms rather than CMIP ones. In addition, the SMS were not designed from scratch but were heavily influenced, together with the service delivery systems, principally by models from the TINA Service Architecture. The use of ODP viewpoints in the documentation of the TINA Service Architecture motivated the adoption the viewpoints in the development approach. The development approach is described here in terms of the main processes of interest, i.e. the modelling the business requirements for the multi-domain context, the modelling of TINA systems as reusable components and the design and implementation of the SMSs that use these components.

4.3.1.1 Business Modelling

This activity used the same ORDIT based techniques used in PREPARE, which were based on the identification of business roles and the responsibilities between them, alongside scenario descriptions. Figure 4-10 is the OMT diagram used to summarise the organisations, their roles and the contractual relationships between the organisations. The contracts were defined by the aggregation of the responsibilities identified between the roles.

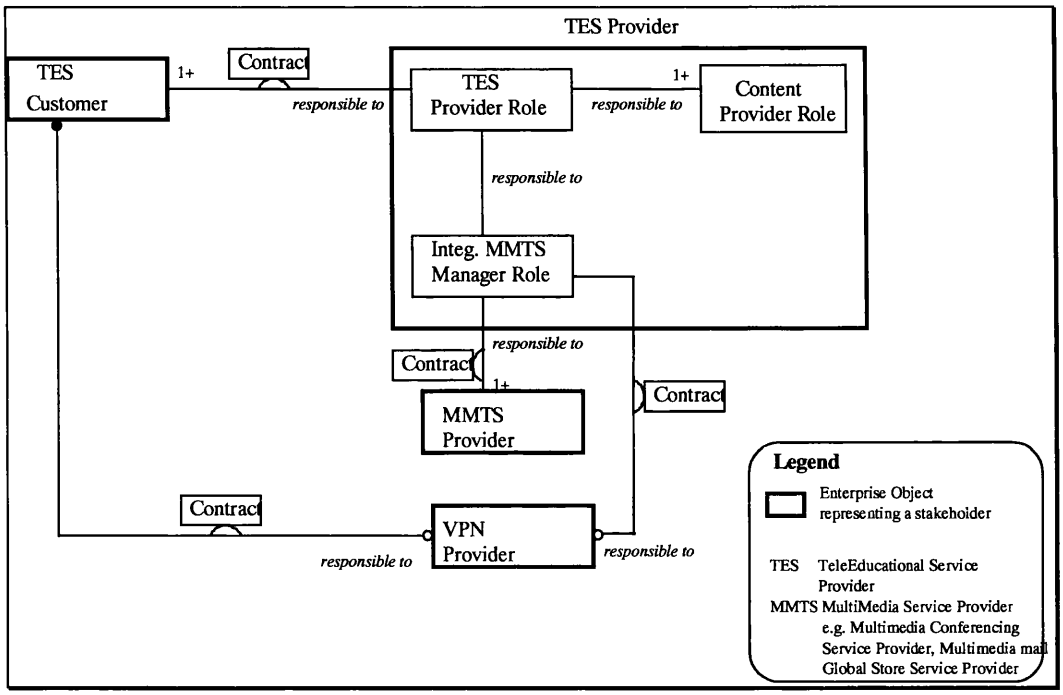


Figure 4-10: Contractual Relationships Between Stakeholder Organisations

The scenarios descriptions were first defined as use case descriptions for the customer, provider and end user roles of the TES stakeholder. Use cases described the interactions of a user role with the multi-domain system as a whole with the aim of performing some task of value to that user. Examples of such use cases were; subscription to the TES, inclusion of a customer network site in a TES subscription, authorisation of a TES end user and the actual use of the service. To assess the inter-domain implications of these use-cases, i.e. the requirements they placed on the different stakeholder organisations in the enterprise model, high level sequence diagrams were drawn up to help define the information that needs to flow between the different stakeholder and roles. These were equivalent to the scenario descriptions performed in the previous two case studies in that they revealed the required inter-domain interactions. An example of such a diagram for the "authorise a TES end user" use case is shown in Figure 4-11.

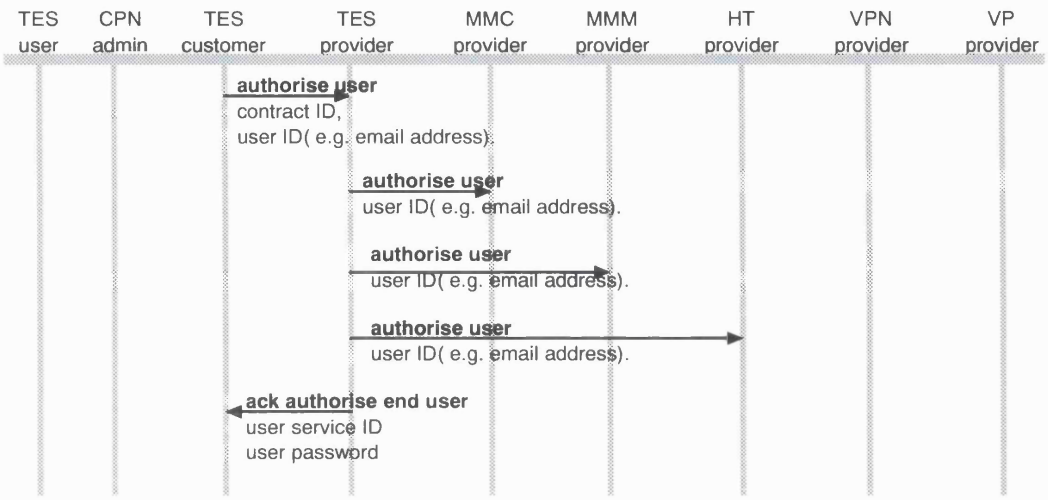


Figure 4-11: Scenario Sequence Diagram Showing Information Flow Between Stakeholders for a Use Case

4.3.1.2 Reuse of Existing Models

Pre-existing management system specifications were analysed to see if they could be reused in meeting these functional requirements. However, in the particular service management areas covered by the requirements, little was available in the way of existing specifications, either from the TMN series of recommendations or from the TMF information agreements. The TINA Consortium, however, had been examining areas of service management in detail as part of its Service Architecture. This provided, amongst others, a generic model for service access and session control. This session model was integrated with a subscription management model for determining which users could access which service from which network terminals and an accounting management model of collecting data on individual user's service usage and transforming this into billing information. This service access and service session model, subscription management model and accounting management model where selected as the basis for common reusable component specifications that could be used in the TES, MMTS and VPN stakeholder SMSs. However, the TINA Service Architecture models assumed only a single provider offers services to customers, whereas the use cases placed requirements on the TES system to integrate

the MMTS offered by other providers into a single service offering. This required the extension of the TINA specifications in order to deal with the resulting inter-domain interactions.

The TINA Service Architecture models were based on ODP concepts, and consisted of:

- An Information Viewpoint model in terms of information object (IO) descriptions in Quasi GDMO together with OMT object diagrams to express the relationships between the objects.
- A Computational Viewpoint model in terms of computational object (CO) textual definitions in ODL together with ODL diagrams showing the client server relationships between objects.

These information and computational models were therefore used as the basis for developing design models that satisfied the requirements presented by the business model and from which the components and systems could be implemented. It was found, however, that the TINA design specification in the form of these two viewpoints was inadequate for this task. This was primarily due to the lack of an explicit linkage between the two viewpoint models, i.e. the mapping between IOs and COs was not presented in the TINA specifications in any clear manner. This prevented both a clear understanding of the system and hid the overall object model needed to implement this system. The first step to resolving this problem was to generate sequence diagrams describing the flow of information between COs. Though such diagrams were present in the TINA Service Architecture, they were presented as selected examples of the application of the models, rather than depicting the general usage of the models. The use case based sequence diagrams illuminated the general dynamic behaviour of the model in satisfying the system's requirements, and in the process clarified the relationships intended between the COs and IOs and their behaviours.

As COs are taken to be units of object distribution, some mechanism was required to map CO definitions to a form suitable for implementation on a distributed platform.

TINA assumes a DPE that provides distribution transparencies for engineering computational objects that implement the COs. However, no practical implementation of the TINA DPE platform implementation was available to the project. Instead a commercial CORBA 2.0 implementation (Orbix from Iona) was chosen as the platform for the SMS. This required mapping between the multiple interfaces of a TINA engineering computational object to the single interfaces of CORBA objects as suggested in [kitson]. This mapping exploited the similarity between ODL and CORBA's IDL, with ODL CO interfaces being mapped to individual IDL interfaces definitions, which were grouped in modules mapped from CO definitions.

4.3.1.3 System Development

As the design of the TINA Service Architecture subscription management component had been presented as a set of IO and CO definitions, and since the relationships between these sets of objects have been clarified through detailed sequence diagrams, the design of the stakeholder systems that use these components used the same modelling approach.

Figure 4-12 shows the OMT object diagram for the subscription management component together with its relationship to the additional IOs (shown shaded) needed to satisfy the multi-domain requirements of the TES SMS. The intention of these extensions was to support the functionality required, while preserving the integrity of the existing component's information model.

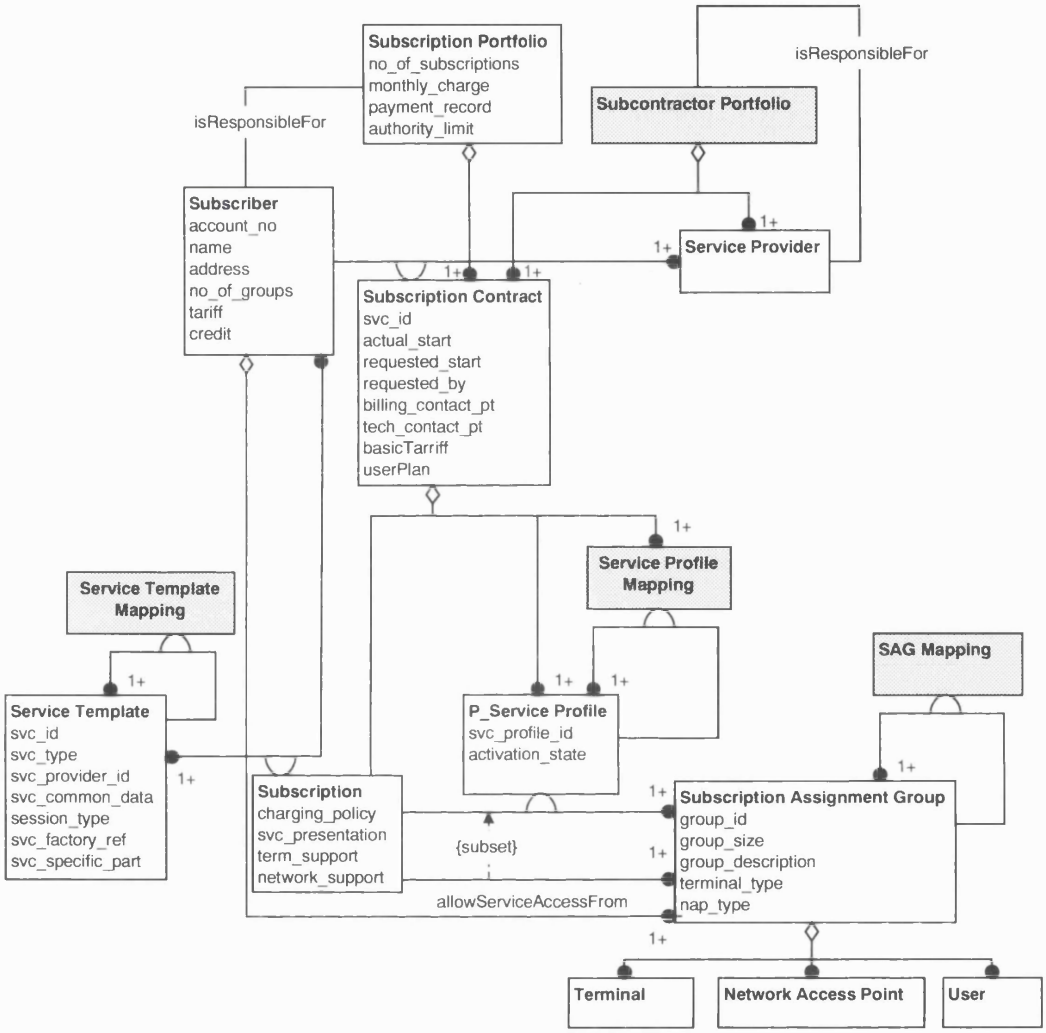


Figure 4-12: Extended Subscription Management Information Model

A similar approach was taken when modelling how the subscription management component's computational model would be applied to the TES SMS design. It was deemed useful to retain as much as possible of the interface definition of the existing COs when developing the extensions required. In this way components designed to interact with the original CO interfaces of the component (SubMgmt in Figure 4-13) could also interact with the extended SMS (SubMgmt* in Figure 4-13) with minimum modification. This was performed simply by designing SubMgmt* as a wrapper for SubMgmt, with the new COs introduced to implement this wrapper

(shown shaded in Figure 4-13) inheriting IDL interfaces from COs in SubMgmt. The SubMgmt* COs provide the functionality needed to interact with SubMgmt components as used in the subcontractor's domains, thus exploiting the same CO interfaces and minimising the complexity of information transformation that needed to be performed.

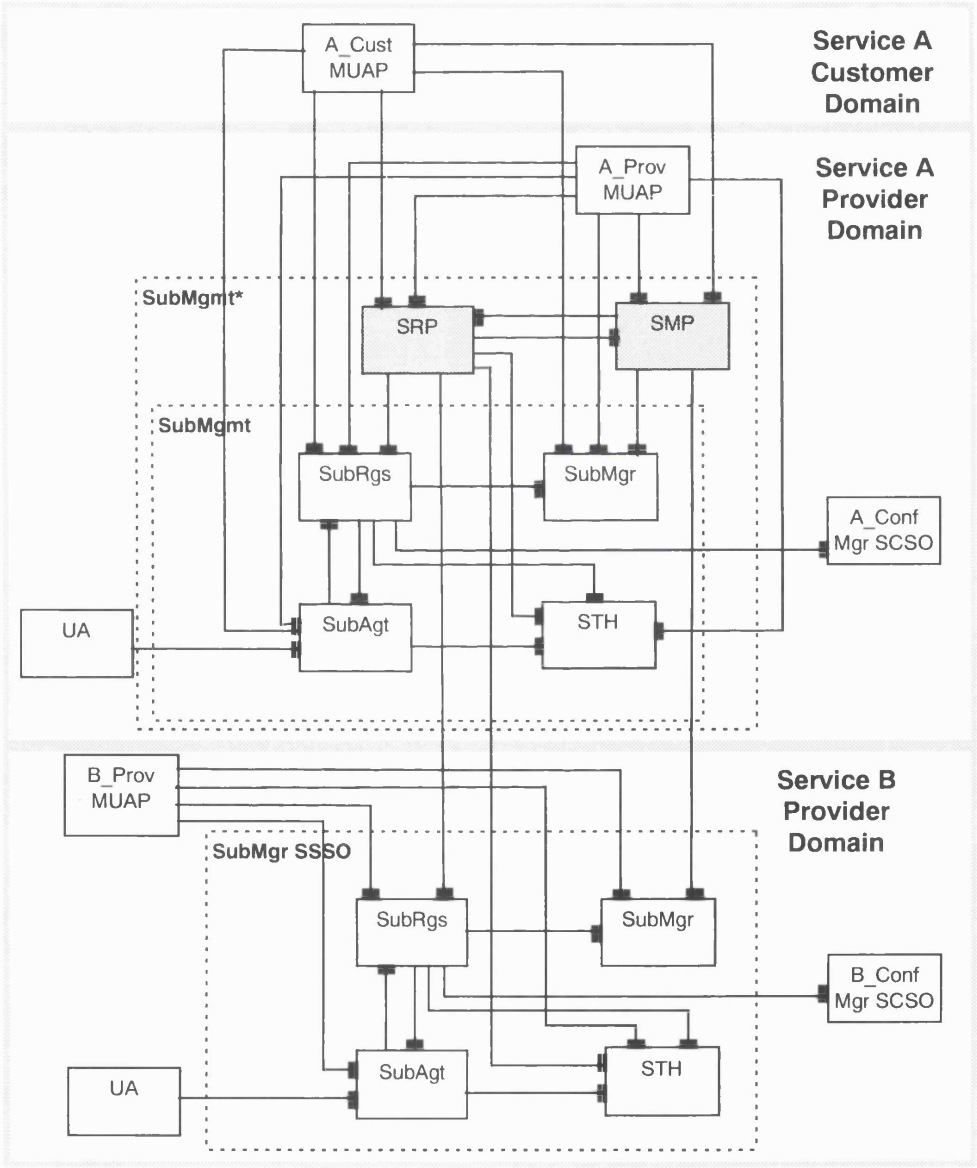


Figure 4-13: Extended Subscription Management Computational Object Model (ODL)

Objects in the original TINA specification, the SubMgmt* COs were documented as a detailed block diagram identifying the specific server interfaces offered by the CO using the IDL interface names. In addition the other COs to which the CO was a client were also identified. An example of this notation is given in Figure 4-14.

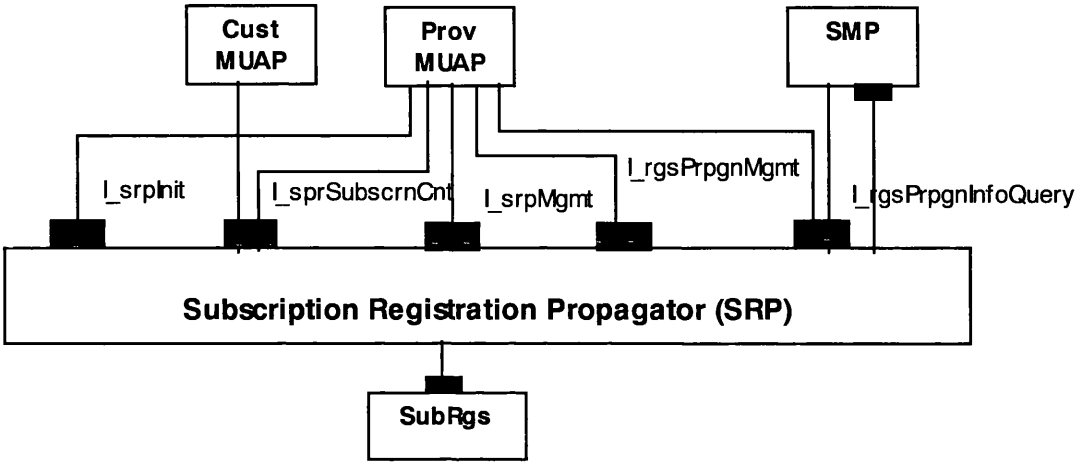


Figure 4-14: Example of Detailed CO ODL Diagram for SRP CO

Such diagrams were accompanied with details of which IOs the CO had responsibility for and descriptions of the functionality provided by the different interfaces. As with the original TINA COs, sequence diagrams showing interface interactions between the SubMgmt* COs were used to develop these interface definitions and clarify which IOs are held in which COs. An example of such a sequence diagram is given in Figure 4-15.

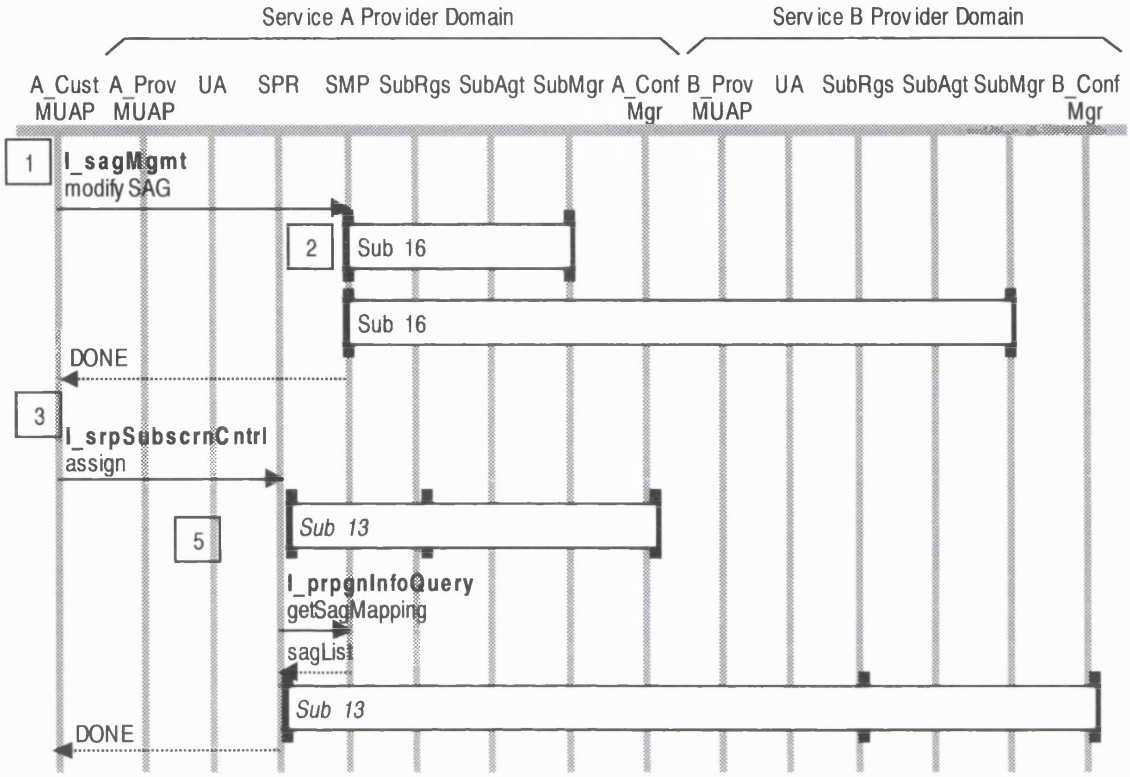


Figure 4-15: Example of Sequence Diagram Showing Interactions between COs

The functionality covered by these sequence diagrams was taken directly from the use case information flows used in the analysis, thus providing a mechanism for ensuring that the requirements were fully met by the design. Figure 4-15 shows the interactions that implement the use case information flows of Figure 4-11. As sequence diagrams showing interactions between multiple COs in multiple domains could easily become large and complex, a nesting notation was used to refer to sequences of interactions that were represented in other diagrams, e.g. the boxes marked Sub_16 and Sub_13 in Figure 4-15. This form of nesting sequence diagrams also simplified the drawing of situations were sequences of interactions were repeated. The boxed numbers referred to accompanying notes that explained each significant interaction in more detail, in particular, referring to their effect on IOs contained within the COs shown.

As well as proving essential in clarifying the behaviour of CO interfaces and their internal operations on IOs, the sequence diagrams were also found to be ideal for producing test documentation. Integration tests performed between components implemented by different developers were specified by defining pre-conditions and post-condition values for IOs at the beginning and end of sets of interactions represented on an sequence diagram. Values were also provided for the parameters of interface operations performed, so that appropriate test harness software could be developed and operated. This was especially important where interactions involved a chain of several COs, and these needed to be tested individually and in small groups before finally being able to test the complete end-to-end interaction. This involved the definition of test cases at a finer level of granularity than those derived directly from the system level use cases.

4.3.2 Evaluation and Results

This case study provides evidence on the usefulness of techniques such as use cases, OMT graphical object modelling and ODP viewpoints as practised by the TINA-C. The reactions of the developers to these modelling techniques was gathered though a group discussion. The discussion was chaired by the author and was driven by the review of the answers participants had given to a questionnaire in the weeks prior to the meeting. This questionnaire elicited views from the developers on the usefulness of the modelling techniques used in: capturing and revising requirements; defining the enterprise model; writing scenarios; designing the components and SMS using the ODP information and computational viewpoints; implementing the system using a CORBA-based engineering viewpoint and testing.

The general structure of ODP viewpoints was applied in this case study. The capture and analysis of requirements was classified as enterprise modelling, the design of both the individual stakeholder's SMS and of the components used within them was conducted using the information and computational viewpoints and the implementation was guided by the engineering viewpoint. The activities used in developing the systems and components are summarised in Figure 4-16.

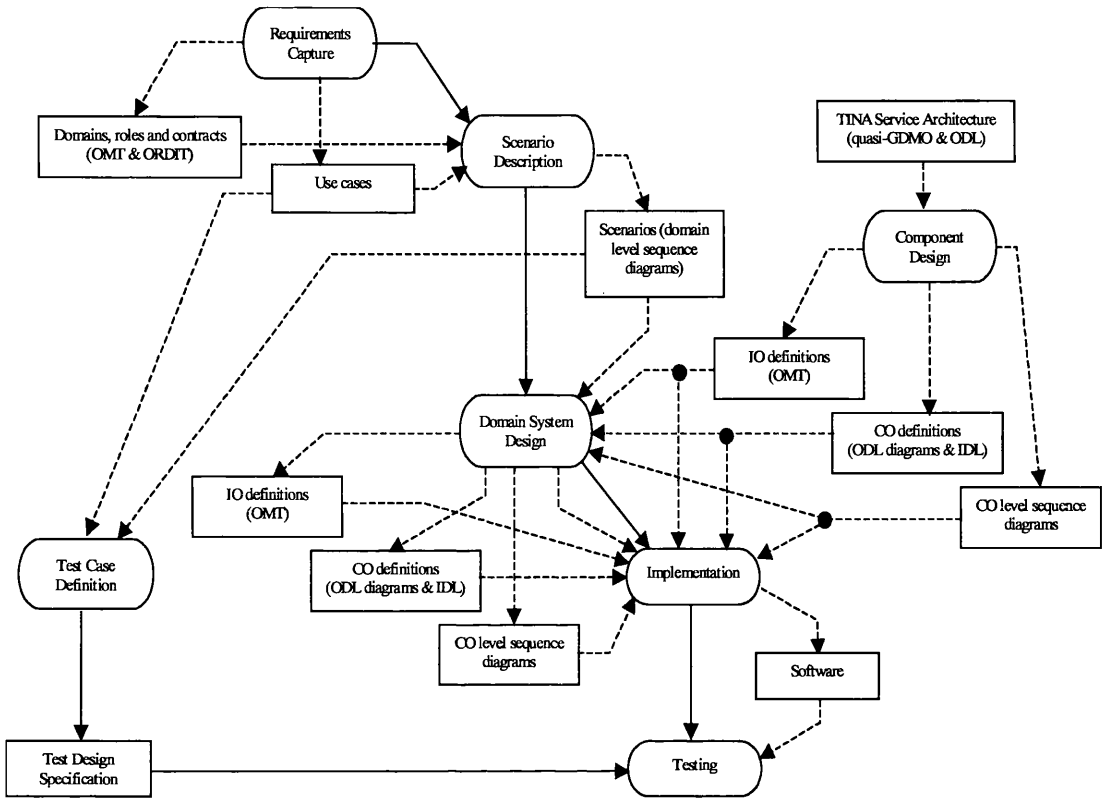


Figure 4-16: Development Process for Case Study 3

The dual requirements capture approach of responsibility modelling and scenarios modelling was retained from Case Study 2. An intervening stage was introduced however, where use cases describing tasks performed by the human roles were used to motivate individual scenarios. In addition, the organisational stakeholders, the business roles they play and the contractual relationships between them were modelled using OMT class diagrams. The reaction of those performing the enterprise modelling was that, though the identification of responsibilities between roles was useful, the refinement into obligations, actions and resources was difficult to perform and relatively unhelpful. This was due to the design being based to a large extent on existing specifications, so that the freedom provided by this top-down approach was not available, and merely served as a consistency check.

The reuse of TINA models was exercised at both the SMS design level and at the level of the components that made up these systems. For both, the ODP information

and computational models were used though several methodological problems were experienced by the developers in using the two viewpoints. The major problem was in relating the two viewpoints to each other. It was found that the notations used in TINA did not provide adequate support for mapping model elements between the two viewpoints. This was found to be necessary in order to gain a complete understanding of the designs being used. The TINA specifications used, therefore, had to be supplemented by interaction diagrams to gain a fuller understanding of the mapping between objects given in the two viewpoints. This was still only an implicit, rather than explicit mapping, and was difficult to maintain when changes were made to models in either viewpoint. As models in the two viewpoints were closely coupled, changes in the information model often resulted in changes in the computational model and vice versa. Similar problems were encountered when developing new designs using the two viewpoints. Modelling using OMT for the information model, ODL for the computational object and sequence diagrams for dynamic models made the use of CASE tools difficult. Where CASE tools were available, their use was found to be very beneficial to the developers common understanding of the design [neilsen]. This was performed using the available set of OMT object, dynamic and functional models rather than the ODP-oriented models used in the rest of the project. The division between the two viewpoints, therefore, was found to be a barrier to the developers' comprehension of a design and made the task of consistency checking an onerous one. Developers tended to be most interested in the computational viewpoint, since this was the one in which interface agreement to other sub-system were defined and which had a major impact on interoperability. The agreement of IDL interfaces was therefore seen as the most crucial collaborative design activity. Other aspects of the engineering viewpoint were not actively addressed as the implementation of location and access transparencies and the underlying communication protocol were provided by the ORB.

From this case study we can conclude that ODP suffers in several respects. The mapping of viewpoint concepts to practical development notations for SMS

development was not sufficiently defined. Also, as discussed in Chapter 3 there is a general lack of consistent guidance on a suitable development process for actually applying ODP modelling constructs to the development of SMS. Developers, therefore, have to define their own process, as was the case in this case study. Finally, consistent with the other conclusions, there is little tool support available for development using ODP viewpoints in commercial CASE tools. Due to the close coupling between the information and computational viewpoints, such tool support is essential for the seamless transition between viewpoints and for automatic consistency checking between them if this technique is to be applied successfully.

4.4 Case Study 4: Developing SMS with UML

This case study is based on the second phase of the Prospect project. This was based on the same tele-education service as the first phase, with the resulting multi-domain SMSs also being demonstrated through user trials. Several trial systems were developed in this phase, each demonstrating a different multi-domain business scenario. These scenarios aimed to show how SMS could be constructed to flexibly support multiple business scenarios and how service management components could be reused in different stakeholder's SMS, across these different business scenarios. A review of the approach taken has been published in [lewis99d], the author's contribution to which forms the basis of the following section. This phase of Prospect was also subject to a more in-depth evaluation of developers' experience of the methodology, performed through a questionnaire. The results are presented in the subsequent section.

4.4.1 Development Approach

The development approach followed was heavily influenced by the methodological experiences of the first phase as presented in Case Study 3. The principle result of this was that two modelling processes were explicitly identified in the development of SMS:

- *Multi-Domain Modelling*: This captures requirements of management tasks involving more than one organisational domain. It therefore focuses on supporting inter-domain interactions.
- *Single-Domain Modelling*: This captures the management system requirements and design for a specific organisation. It therefore focuses on intra-domain interactions.

These processes are not independent, so the approach taken supported the alignment of requirements and interface definitions between a multi-domain model and the related single domain models.

In addition to these two modelling processes, the approach taken also explicitly addressed the modelling of components. Components were treated as separate entities from multi-domain or single-domain systems on the assumption that they may be developed by third party vendors and will, therefore, have distinct development life-cycles. This approach had to support the introduction of separate component models into the development of multi-domain or single-domain systems. This case study, therefore, provides a close match to the generic SMS development process model of Chapter 2.

The recognition of the presence of distinct development stakeholders, which was reflected by the collaborative nature of the project, highlighted the need to support as much as possible the communication of models between different developers. The emerging UML standard was therefore selected as the primary modelling notation for the development approach of this phase of Prospect. This decision was justified by the broad range of modelling constructs it supports including ones similar to those familiar to developers from the first phase of the project. The support for UML by commercial CASE tools which were used in the project, such as Paradigm Plus and Rational Rose, was also a major motivating factor. To fully support the development cycle of the required SMS, detailed design specifications and specifically interface definitions had to be in technology specific languages, in this

case IDL. The CASE tools used already supported mappings from UML class definitions to IDL.

The development process used aimed to provide a common approach to iterating through the development of management systems, whether they were multi-domain systems, single-domain systems or components. As proposed by this thesis, in this case study it was expected that, by following a common well understood process and notation regardless of the type of system being implemented, communication between developers of these different types of systems would be facilitated. The process adopted can be decomposed into the following steps:

- Definition of the system business model, identifying the business stakeholders and roles together with their responsibilities and obligations to each other.
- Functional requirements capture by use case analysis.
- Identification of system information in terms of objects and their relationships.
- Functional decomposition of the system into sub-systems, including identification of pre-existing, reusable components, the definition of external interfaces and interfaces between sub-systems.
- Definition of distributed platform structure and required services.
- Definition of test specifications.
- Implementation and integration of components.
- Testing of sub-systems, sub-system integration testing and testing of external interactions.

The process and notation is discussed in the section in terms of their application to multi-domain system modelling, component development and single-domain SMS development.

4.4.1.1 Multi-domain System Modelling

The enterprise model for a specific business scenario was represented using UML object diagrams. The objects in the enterprise model diagrams were instances of classes from a general enterprise model, shown in the class diagram in Figure 4-17. In this general model classes representing roles and stakeholders are differentiated by their class stereotypes. The general enterprise model defined a set of roles and stakeholder that were thought likely to be present in multi-domain, open service management scenarios. However, this was principally performed to clarify the context of Prospect's work, and other general enterprise model classes could be equally valid in different situations.

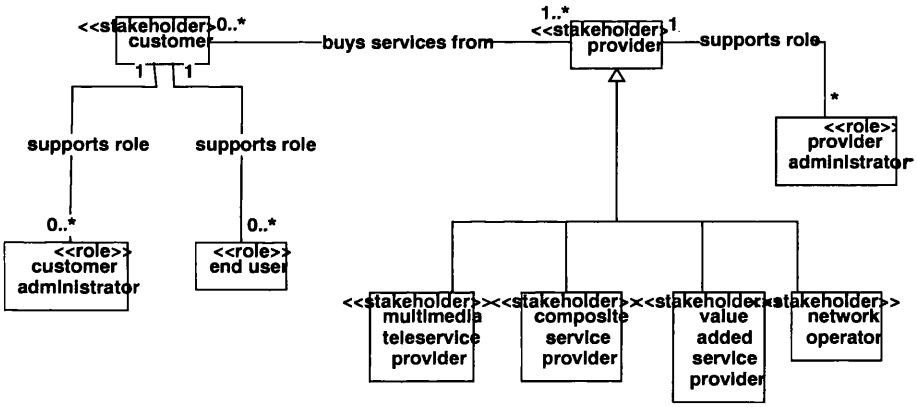


Figure 4-17: UML Class Diagram Showing Roles and Stakeholders Used in Prospect Trials

To provide a more detailed context for the subsequent definition of use cases, the relationship between the roles and organisations prior to the trial was also described. This took the form of statements of contractual responsibilities between the different stakeholders that could, in commercial scenarios, form the basis of, or be informed by, legal contracts between the parties concerned. These contractual responsibilities were represented as associations between stakeholder objects. A further breakdown into obligations with mappings to activities and resources using the ORDIT technique was not attempted.

Use cases at the multi-domain system level defined what the system as a whole needed to do in terms of useful interactions with actors that define the system's environment. Figure 4-18 shows an example of a UML class diagram for one of the multi-domain trial systems developed. These actors represented instances of the role class stereotypes from the general enterprise model for the multi-domain system. The use cases descriptions were stated in the form of text, with sections defining the use case pre-conditions, the use case itself and the use case post conditions. The preconditions present the state of the multi-domain system, from the point of view of the actors, and would typically be related to the post-condition of other use cases. The use case body described, as a sequence of steps meaningful to the actor, the interactions that were performed with the system in order to complete some useful task.

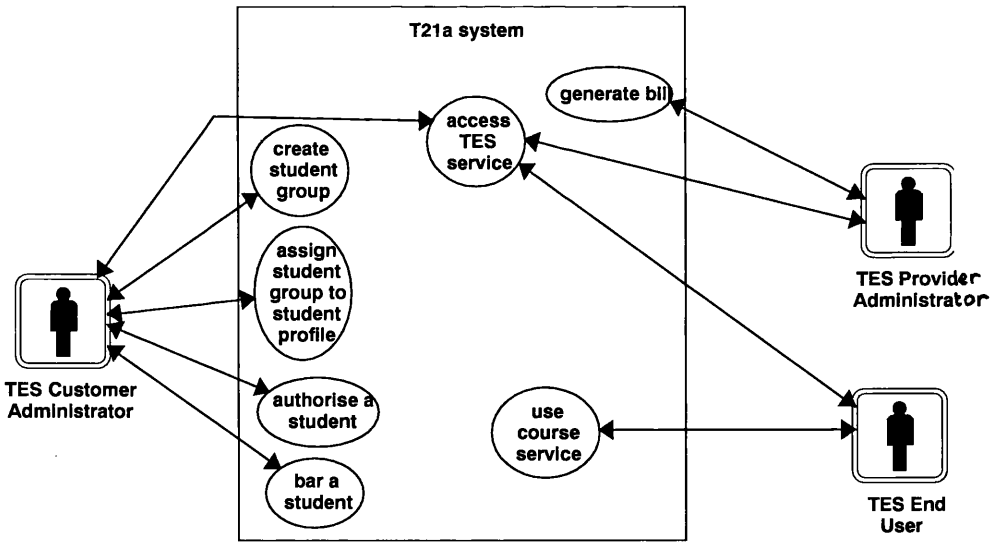


Figure 4-18: UML Use Case Diagram for Prospect Customer Management Trial

To analyse the inter-domain interactions within such a multi-domain system, the individual use cases were refined to describe the inter-domain interactions they required. This step was informed by the responsibilities between different roles and stakeholders in the enterprise model. Refining the multi-domain use cases in this way enabled the identification of use cases for individual domains, i.e. for the single-

domain systems that made up the multi-domain system. A similar set of use case diagrams showing the decomposed, single-domain use cases, could then be produced as input to the single-domain requirements discussed in Section 4.4.1.3. However, use case diagrams in UML do not support direct interaction between use cases, so use case diagrams could not be used to show the full chains of interactions between the single-domain systems that make up the multi-domain one. Instead, high-level UML sequence diagrams were also used to show information flows between the multi-domain system actors and objects representing individual domains, in a similar manner to the previous case study.

The definition of the information was involved in these inter-domain interactions was based on the requirements embodied in the multi-domain use case definitions. However, developers also took information definitions from existing standards and from existing components that were likely to be used in the systems implementation.

4.4.1.2 Component Modelling

The components used in this phase of the project were modified versions of the ones based on TINA Service Architecture specifications that were implemented for the first phase. Components were re-modelled in UML using CASE tools. Use cases were introduced to the component model to define the actors that would interact with a component and to define their interactions with the component. A use case diagram for the Subscription Management component described in Case Study 3 is given in Figure 4-19.

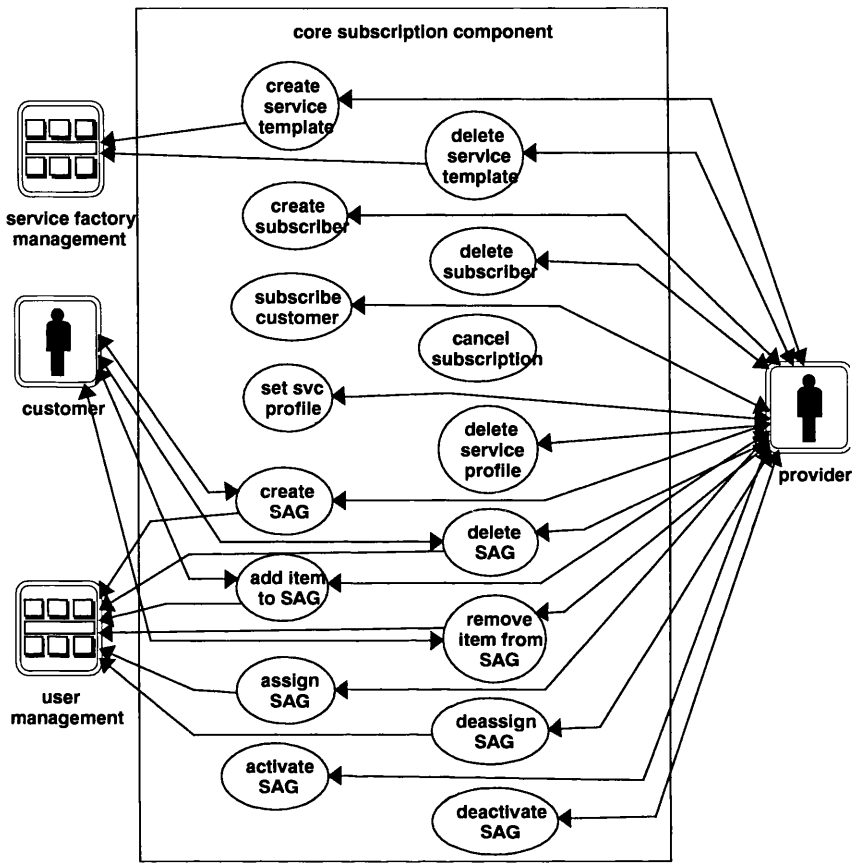


Figure 4-19: UML Use Case Model for Prospect Subscription Management Component

Note that some actors represent human users while others represent systems that may be other components that are either abstract or represent existing components, such as ones from the TINA Service Architecture.

The design of components was developed using UML class diagrams. These represented the results of both the information modelling activity and the functional decomposition activity. Outputs from both activities were integrated on the same diagrams but were differentiated by stereotypes for information objects (IO) and computational objects (CO) respectively reflecting the design's TINA origin. Figure 4-20 shows the top-level class diagram for the Subscription Management component. UML component diagrams could have been used to identify the different interfaces of computational objects and their relationships. Instead, however, the

details of the interfaces were modelled as classes grouped in diagrams representing a single CO. This facilitated both the automated generation of IDL by case tools and the maintenance of consistency with interaction diagrams, features not directly supported by component diagrams. It meant, however, that the collection of interfaces in a CO construct was not explicitly represented anywhere in the UML model.

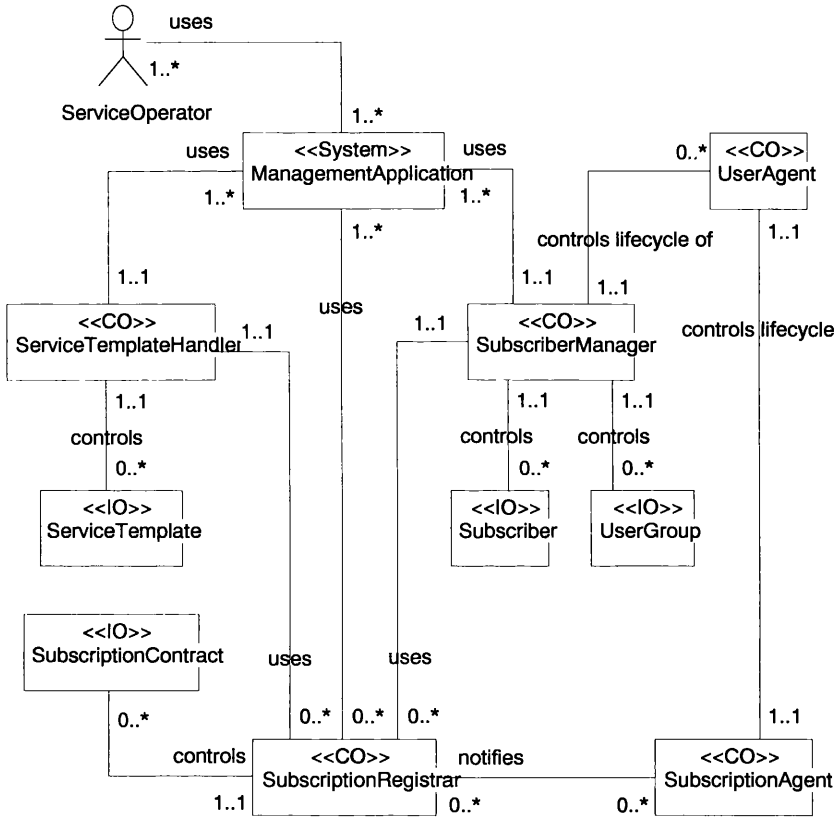


Figure 4-20: Top-level UML Class Diagram for Subscription Management Component Design

Figure 4-21 shows an example of how class diagrams were used to define the interfaces for one of the computational objects shown in Figure 4-20. The computational object, Subscriber Management, has eight IDL interfaces, two of which have been inherited from more general interfaces intended for managing a computational object's lifecycle (i_CoInit) and administrative state (i_CoMgmt).

The interface's operation parameters are not shown in this figure, but were also modelled in the CASE tool used (Rational Rose in this case).

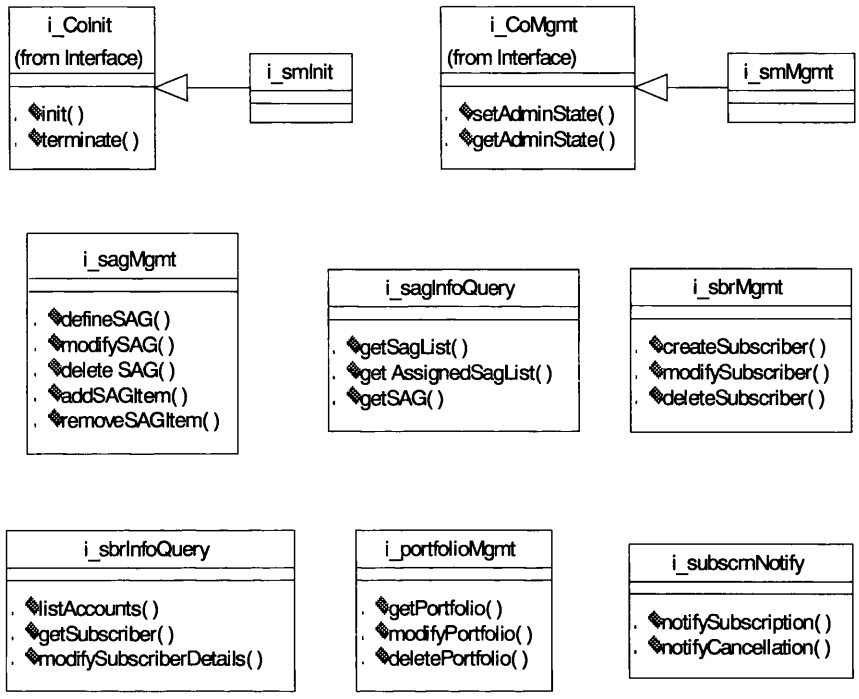


Figure 4-21: UML Class Diagrams Showing the Interfaces to the Subscriber Manager CO

To fully describe the component's behaviour for the benefit of the implementers, its dynamic operation had to be modelled. This was typically performed by defining the interactions between the computational objects and with actor systems, based on individual use case descriptions. An example of such an interaction diagram is given in Figure 4-22, which shows the interactions between entities in terms of IDL operations on their interfaces. This example shows the CO behaviour required by the "Create Subscription Assignment Group (SAG)" use case shown in Figure 4-19. Note that in this diagram only the Subscriber Manager and the Subscription Agent entities are part of the Subscription Management component. The management user application (MUAP) is the design level representation of the application used by the Provider Administrator actor identified in the use case model, while the User Agent

object is part of the User Management system actor also identified in the use case model. These are necessary since the dynamic behaviour of a component can only be fully described by including its interactions with its environment.

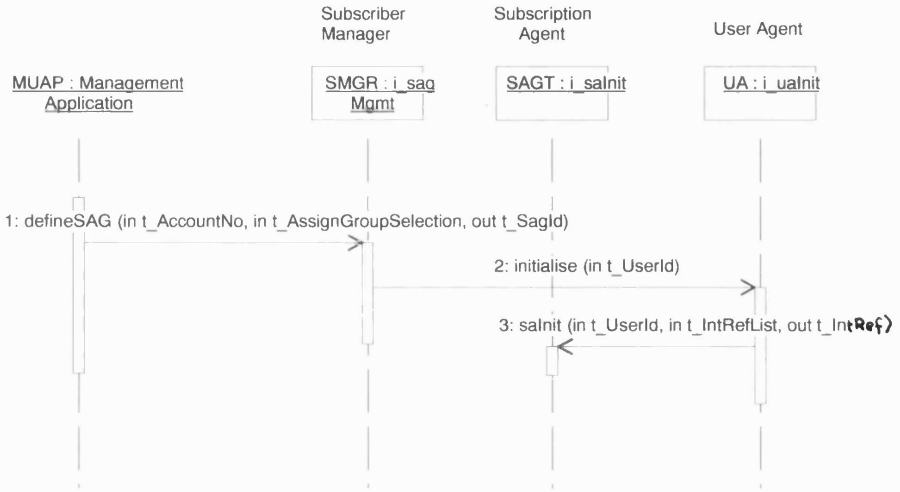


Figure 4-22: UML Interaction Diagrams Showing Subscription Component Behaviour for the Create SAG Use Case.

4.4.1.3 Single-Domain System Modelling

The development of SMS for Prospect was performed at the level of an organisational domain, i.e. the SMS contained all components and subsystems operated by a single organisation. In determining a system's requirements, in addition to inter-domain sequence diagrams, multi-domain system use cases were used/ These were decomposed into linked sets of use cases specific to the constituent single-domain systems. This helped to identify where the same functionality was required in different domains. Where such common functionality could be provided by a reusable component, these components and their relevant use cases were included in the diagrams expressing the multi-domain to single-domain use case decomposition. An example of this is shown in Figure 4-23.

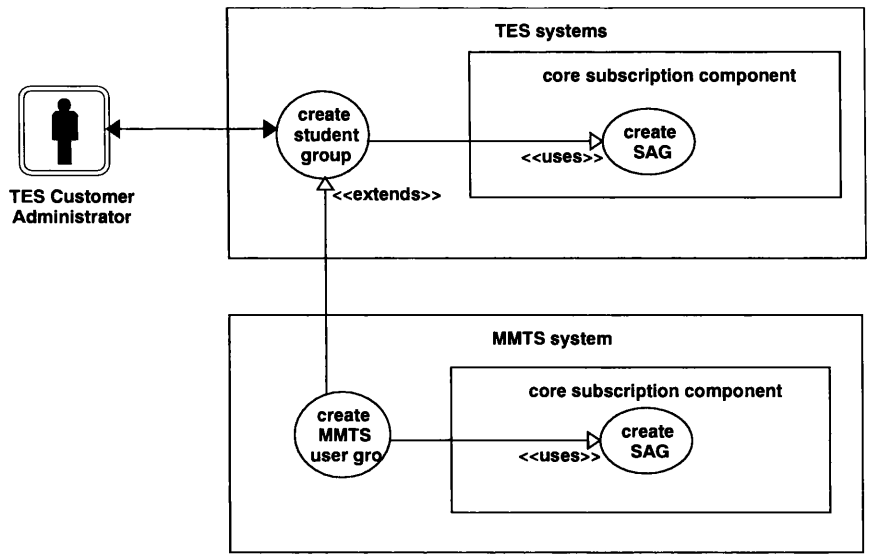


Figure 4-23: Multi-domain Use Case Linkages Supported by Component Level Use Cases

In this way, places where components could be reused in different domains could be clearly identified. This analysis also aided in the identification of the areas where reusable components did not satisfy the requirements of the domain's use cases, and where, therefore, the development of additional domain-specific sub-systems was required. Such sub-systems were modelled using the same notational techniques as those used for developing and describing components

4.4.2 Evaluation and Results

The development methodology used in this case study combined an iterative, use case driven development process with UML as the modelling notation. This has been applied effectively for multi-domain management system analysis, single-domain system development and component development. By using the same basic methodology for all of these activities communication between the different developers, e.g. component developers and component re-users, was eased. Figure 4-24 gives a summary of the common development process.

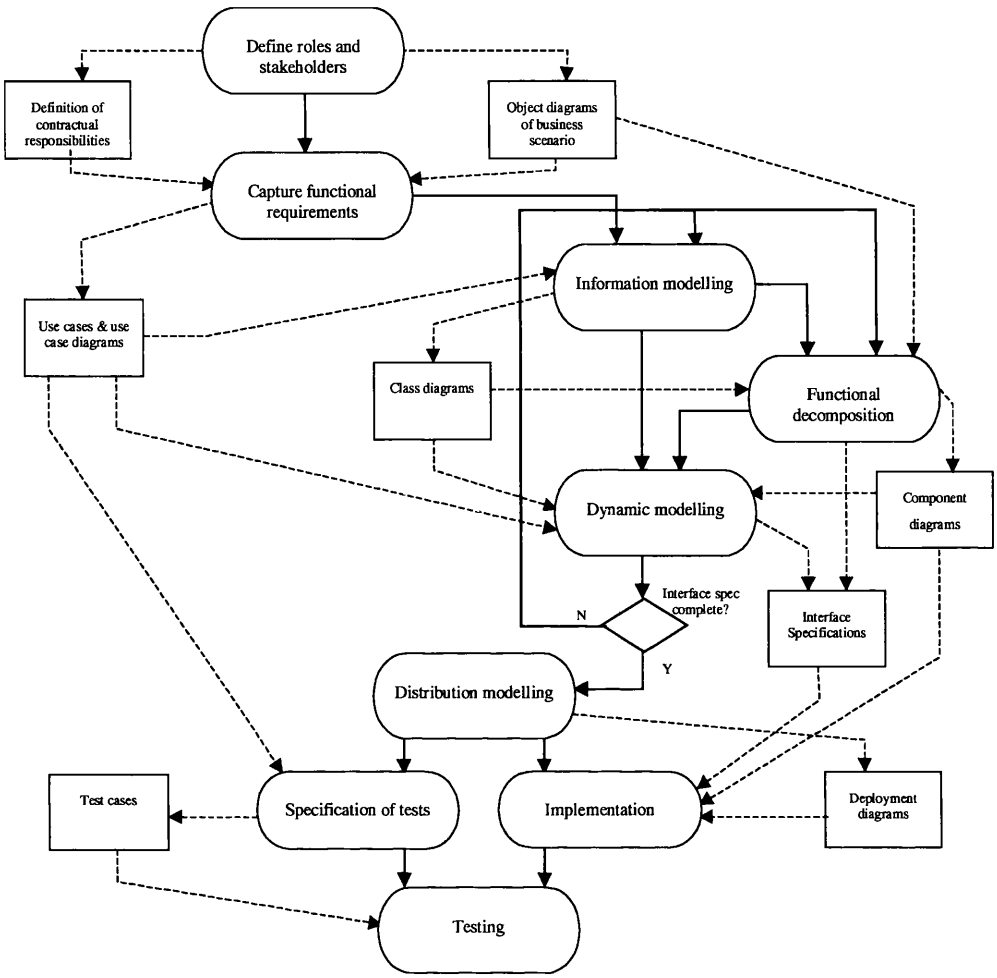


Figure 4-24: Development Process for Case Study 4

It should be noted that some models are used in several subsequent steps. Also note that the information modelling, functional decomposition and dynamic modelling steps are closely coupled and may undergo several iterations before arriving at a set of completed interface specifications.

The approach followed in this case study benefited from the experience of the previous case study and also from the emergence of the UML notation, which was bolstered by strong tool support. It made the distinction between the application of a common development methodology to the analysis of multi-domain systems, to the development of complete SMS and to the development of reusable management components. This case study, therefore, partially emulated the business model for

SMS development proposed in Chapter 2. The latter two applications of the methodology are analogous to those conducted by the SMS Developer and the Software Vendor respectively. This case study is therefore important in determining the effectiveness of a common methodological approach in communicating between these stakeholders.

The use of UML brought several advantages. Its coverage of a wide range of modelling diagrams meant that a tool that supported UML would provide much better coverage of the models needed compared to the tool support for ODP-based modelling experienced in the previous case study. Also the stereotyping mechanism of UML allowed its modelling constructs to be tailored to the SMS development problem domain. This was used, for example, in the definition of stakeholder and role stereotypes for business level diagrams. In the long term UML stereotyping provides a path to standardising modelling constructs for this domain and influencing tool developers to support them.

Use cases were used more widely in preference to scenarios due to their better-defined semantics. Use cases were used in the three different development areas for showing how the SMS and components under development interacted with their environment. However, this was to the detriment of the dynamic modelling at a high level abstraction, with dynamic modelling being mostly focussed at the design stage.

Several shortcoming of UML were identified in this case study. Use cases lack a relationship where use cases in one system interact with use cases in another system, a facility that would be useful in decomposing multi-domain system requirements into single-domain system ones, and similarly in matching single domain system requirements onto interacting components. Also, though multi-interface computational objects can be modelled as components in UML component diagrams, these cannot be used as communicating entities in interaction diagrams, restricting support for multi-interface distributed components. These problems point to improvements required in the semantics of UML rather than just ones that can be implemented through stereotypes

The above observations are based largely on the author's experiences in analysing and using the methodology proposed for this case study and from discussion with the developers. In addition, a questionnaire was used to get a more structured view of the developers' experiences, against which the above observations can be compared. The questionnaire was designed using the goal-question-metric approach described by [basili] and structured using questionnaire writing techniques described in [oppenheim]. The questionnaire to target those involved in developing whole domain management systems, developing individual subsystems for a specific domain management system and/or developing reusable components applied in several subsystems. Questions eliciting feedback from these activities were of two types. The first attempted to gauge the usefulness, if present, of different modelling constructs for a specific development activity. Responses were requested on a scale of:

Essential =5, Mostly Useful =4, Generally Useful =3, Partially Useful =2, or Not Useful =1.

The second type of question attempted to discover the extent to which errors in the development process were due to omissions in, errors in, inconsistencies in or misunderstanding of other models. Responses were requested on a scale of:

Always=5, Usually=4, Sometimes=3, Rarely=2 or Never=1.

For both types of questions respondents could also register a "don't know" for individual categories.

The questionnaire therefore contained six sections as follows:

- The first section allowed the developers to identify which components, whole domain systems or constituent sub-systems they were involved in developing and for which parts of the development cycle, i.e. analysis, design, implementation and/or testing/integration. This was intended primarily to clarify in the mind of the respondent which development experiences they were referring to in their answers. This information was also available for any correlation needed between

groups working on different systems and differences in their responses those of others.

- Section two was to be answered by component developers. It addressed the component's analysis specification as applied to component design and implementation (Q2.3) and its design specification as applied to its implementation and testing/integration (Q2.7).
- Section three was to be answered by whole domain system developers. It addressed:
  - The multi-domain analysis specification as applied to system design and implementation (Q3.1).
  - The design specifications of interoperable systems as applied to the systems design, implementation and testing/integration (Q3.3).
  - The analysis specification of reused components as applied to system design and implementation (Q3.6).
  - The design specification of reused components as applied to system implementation and testing/integration (Q3.8).
- Section four was to be answered by sub-system developers. It addressed:
  - The analysis specifications for reused components, the encompassing system and other interoperating sub-systems for sub-system design and implementation (Q4.1).
  - The design specifications for reused components, the encompassing system and other interoperating sub-systems for sub-system design, implementation and testing/integration (Q4.3).
- Section five was to be answered by everyone and addressed the CASE tools used for different modelling activities and the communication techniques used between developers at the different stages of development, i.e. meetings, telephone, email, multi-media conference or exchange of revised specifications.

- Section six was also to be answered by everyone and requested personal information from the respondent including the level of experience with modelling techniques and tools used.

Completed questionnaires were received from fifteen developers. It was found that respondents had had difficulty in completing questions on the frequency of problems related to different specification parts due to poor phrasing of the question. This data was therefore not analysed further. The mode response for each modelling technique addressed by each question is given in Appendix 1 (Section 8.1). If two adjacent values contained the same mode total the mode is presented as the median of those two values. If the mode value was present in two unadjacent categories or in more then two categories then no mode value is presented. Though the usefulness measure is not an ordinal scale the mean is also presented to give a further indication of the spread of responses. Uncompleted and "don't know" responses were not included in the mode and mean calculation, with the number of responses counted given on the category entries of the y-axis.

The responses largely reinforced the general observation made above, though the small size of the sample does not allow any strong conclusions to be drawn. Use cases were found to be generally the most useful both for the design of the item analysed and also for understanding the items that needed to interoperate with the item under analysis. Modelling constructs related to role and responsibility were generally not rated as very useful. Significantly, the general textual description of components and systems were rated highly for design, indicating that they provided an aid to understandability that was not present through the other more formal UML constructs.

For the design stages sequence diagrams and collaboration diagrams are highly rated supporting the use of scenarios and use cases as the focussing thread though the different development stages. For the implementation and testing/integration stages the core object specification were seen as essential, though the specification of the

information content of interface operation parameter types together with the CORBA Naming Service naming scheme were also rated highly.

4.5 Case Study 5: Developing SMS with Integrating Business Process Modelling and Component Reuse

This case study is based on work conducted in the FlowThru project. This project had the express aim of investigating development techniques for the integration of management systems that implemented management business processes from reusable management components. This case study therefore provided an opportunity to explicitly analyse the development process needed for such development in addition to observing and evaluating its application. This case study also represents an explicit attempt to address the generic SMS development process model laid out in Section 3.5.4. For this project the author proposed a reusable component modelling approach and an open approach to the mapping of business processes to open reference points based on existing standards. These proposals were published in [lewis99c], from which the descriptions of the development approaches given below are based. The next section also provides examples of how these proposals were applied in the development of management systems within the project. The reusable component modelling approach is applied to an evolution of the same subscription management component used in the previous two case studies and is based on contributions by the author to experiences published in [lewis99b][wade99]. The application of the open business process modelling approach and its integration with the component modelling approach is taken from a portion of the FlowThru system development addressing the integration of service ordering and ATM network planning and configuration as documented by the author in [lewis99e].

4.5.1 Development Approach

The overall development approach proposed for this case study built heavily on the results of Case Study 4, in particular in the use of UML as a notation, the use of use

cases for both systems and component requirements specification and the use of roles and responsibilities in capturing single and multi-domain system requirements. The novel techniques applied were, as mentioned above, the modelling of components specifically for reuse and the use of business process models for analysing system requirements and mapping them to open reference points. These techniques and their application in the case study are presented in the following two sections.

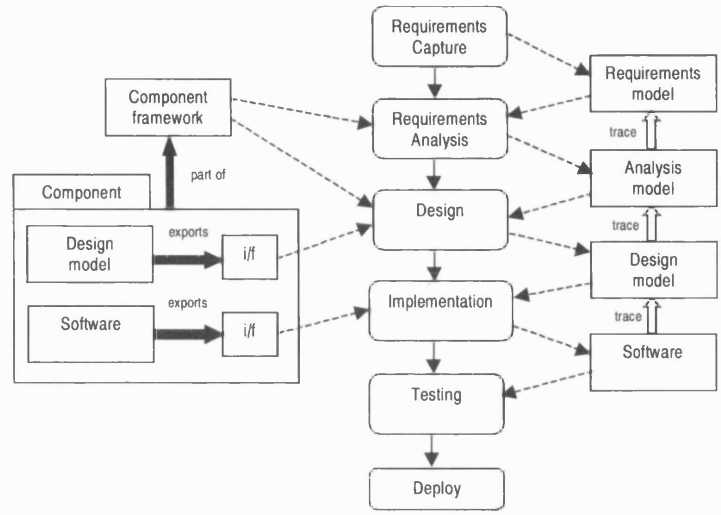
4.5.1.1 Reusable Component Modelling Approach

The approach to modelling components for reuse assumed a development activity similar to that described in Section 3.5.2. Reusable components are typically presented to system developers as sets of libraries, i.e. as a set of software modules and the definition of the individual operations they provide. In terms of the generic development process described in Section 3.5.2, the component is presented in terms of its design model and software. This may cause problems in the development of systems that reuse the component, since any changes required to accommodate the reuse of components are only likely to become apparent during the design process, therefore possibly countering aspects of the system's analysis model.

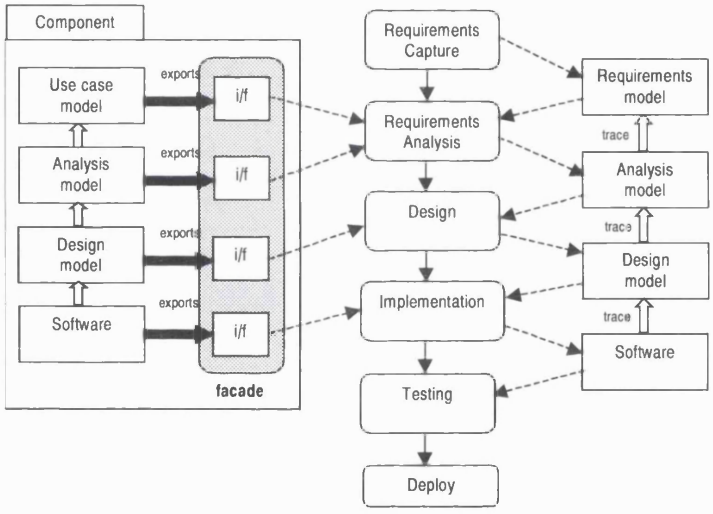
As described in Section 3.5, components are often part of a framework. The framework may be general, e.g. CORBA Services, or aimed at a particular problem domain, e.g. the TINA Service Architecture. In either case, the framework will provide some high level architectural and technological guidance on how components can be integrated together and how they can support the development of a system. Such frameworks are often considered at the analysis stage to ensure that the system's analysis model is structured in a way that will accommodate the inclusion of the framework's components at the design stage. This situation is depicted in Figure 4-25a. However, frameworks typically only give general guidance on the use of components. The suitability or otherwise of individual components in satisfying requirements still needs to be considered in the design activity.

For SMS development, such a typical component reuse situation is difficult to standardise because, as described in Section 3.5, there is no commonly accepted framework that supports a suitably wide range of components. The development guidelines for component reuse presented here are motivated by the absence of such a framework. As such, they attempt to provide guidance on how components can be specified in a more self-contained manner that is easily understood by those performing the analysis of the system. In this way, decisions about reuse can be made based on the suitability of individual components rather than on a wider assessment of the suitability of an entire framework. The approach is aimed at making decisions based on architectural and functional aspects of a component rather than its implementation technology. A component's technology is treated as an orthogonal issue, with heterogeneity handled primarily through the employment of suitable gateways.

The approach is derived from that described in [jacobsen97] and outlined in Section 3.1.1. The basis of the approach is that components are not presented just as units of design and of software within an encompassing framework. Instead, they should be packaged together with the requirement statement and analysis model of the component for presentation to potential reusers. If the modelling techniques used for the requirements capture and analysis modelling of the component are similar to those used for modelling the system in which it might be included, then it becomes much easier for an analyst to assess whether the component is suitable for use in the system. In addition the system's analysis model can directly import the analysis abstractions of the various components it reuses, easing the task of requirements analysis and ensuring, at an early stage, compatibility between components and the system requirements. This analysis model-based reuse approach is depicted in Figure 4-25b.



a) Conventional (design model level) component reuse



b) Analysis model level component reuse

Figure 4-25: Differing Approaches to Component Reuse

The presentation of a component for reuse in this way is known as a facade. A facade presents the re-user of a component with only the information needed to effectively reuse the component, while at the same time hiding from the re-user unnecessary design and implementation details about the component. The facade, therefore, consists not just of reusable code and the design model, but also the requirements statements and analysis model relevant to the design model exposed by the facade.

A component may present several different facades, possibly aimed at different types of re-users, i.e. a black-box reuser such as an SMS developer using a Software Vendor product or a white-box re-user such as an SMS Developer reusing code from previous projects. A component may have various releases of a facade to reflect the evolution of the component. The usefulness of the facade is strengthened if there is clear traceability between the different models, so that within the facade re-users can easily determine which parts are useful to them by matching facade use cases and analysis objects to their requirements and tracing to relevant design model elements. Obviously, the construction of a facade from the internal development models of a component will be greatly eased if the same type of modelling approach was used for developing the component in the first place. Also, traceability in the facade will be greatly eased if the models of the underlying component are strongly traced.

One of the problems raised from examination of the previous case studies was that the boundaries between the different development activities were not always well defined, especially between requirements capture and analysis and between analysis and design. This meant that the level of abstraction used in the models resulting from these activities varied, making it difficult to define traceability mechanisms between the different models. Defining the structure of the different development models was therefore essential to applying useable traces between them.

As use cases had already proven effective for SMS and component development in the previous case studies, Jacobsen's suggestion of using use cases for the requirements model and the closely related robustness model for the analysis model was adopted.

The UML representation of these stereotype classes is suggested in [jacobsen97] and recently proposed to the OMG as an UML extension in [ad/97-08-06]. A slightly different representation available with the Paradigm Plus CASE tools is shown in Figure 4-26.

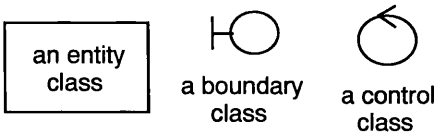


Figure 4-26: Analysis Objects Stereotype Notation

In mapping use cases to an analysis model for a facade, entity objects will be typically derived from noun phrases that occur in one or more use cases. Boundary objects are identified by any interactions between the users and the system. Control objects can initially be allocated per use case, and consolidated as functional commonalties are identified between use cases. Interaction diagrams are often required at this stage to detail the lifecycle of objects and dynamic aspects of the relationships between them. Refining the analysis model into the design model for a facade takes into account design level issues, such as scalability, load balancing, platform dependencies, database schemes etc. Again, strong tracing between objects in the analysis model and the design model is required.

Though it is preferable to generate a facade from a component model of the same structure, it is not a requirement for the component to have been originally developed and documented using an OOSE-like process. Indeed, part of the benefit of facades should be that it can hide the internal model if necessary. For instance if a component has been developed and documented using ODP viewpoints the following mappings may be used to reverse engineering the ODP model into the facade format:

- Roles in the enterprise viewpoint map onto actors in the use case modelled in the facade.
- Computational objects from the computational viewpoint may map onto control objects in the analysis model.

- Computational object interfaces may map or be grouped into boundary objects in the analysis model.
- Information object may map to entity objects in the facade's analysis model.
- If some sequence diagrams have been used to clarify the interactions between computational and information objects, then these might form the basis for reverse engineering use cases, otherwise use cases should be based on and be consistent with the component's requirements.

Such an ODP model to facade mapping was employed as a consistency check in the application of this technique to the subscription management component, as in the following section.

4.5.1.1.1 Application of Reusabie Component Modelling

The facade modelling approach was applied within FlowThru to the integration of components in three separate Trial Business Systems demonstrating different business process areas from the TMF's Telecoms Operations Map. The components from which these systems are constructed are from different EU funded projects, and as such they were originally developed using a variety of notations. To validate the reusable component modelling approach described above the specifications of these existing components had to be recast as facades.

The specific experiences presented in this section are from the development of a facade for the subscription management component that was developed in the previous two case studies. The original specification of this component, as used in Case Study 3, was structured in terms of ODP viewpoints, principally the information and computational viewpoints. In generating the facade for this component the use cases developed for it, as described in Case Study 4, were used to generate an analysis model that would replace the corresponding high level design model shown in Figure 4-20. This was then mapped to the existing design documentation consisting primarily of IDL definitions. The overall aim was to be able to export the facade in a form that presented each of its constituent models and

the traces between them. An HTML-based approach was therefore taken in order to allow traces to be implemented at hyper-links. The Paradigm Plus CASE tools was used for generating use case diagrams, analysis model diagrams and design model diagrams. Though Paradigm Plus supported hyper-textual style links between different diagrams, this was not a suitable solution for widely publishing the facade, whether this is done in the public domain or within a software development organisation. This was partly due to the cost involved in having Paradigm Plus available to everyone who might want to browse the model, and partly because the alternative precluded for the possibility of a similarly structured facade being generated by other tools.

To generate the facade, noun-phrase identification was used on an initial pass of each use case description to elicit the major entity objects and their relationships. The use cases also helped identify a boundary object for each interaction between the system and each external actor. Control objects were required whenever there was a need to create, destroy, update or otherwise manage an entity object or closely linked group of such objects.

The subscription management component produced entity objects such as Service, Customer Account, Service Level Agreement (SLA), Subscription and Subscription Contract, a set of boundary objects, and four major control objects: Service Manager, Customer Manager, Subscription Manager and SUG Manager.

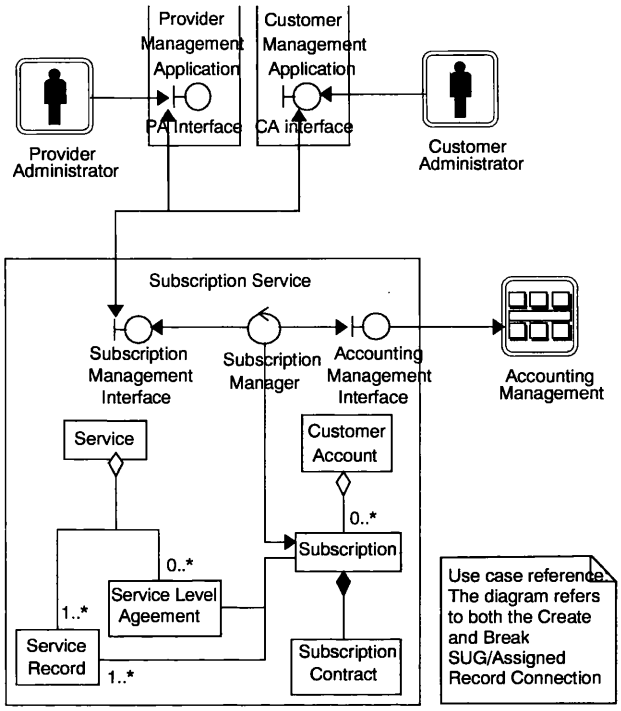


Figure 4-27: Analysis Object Diagram for Subscribe a Customer to a Service Use Case

The objects were then modelled as analysis class stereotypes in UML use case diagrams since this was the only diagram type in which Paradigm Plus rendered the analysis objects' stereotype notation. As use case diagrams are more commonly used to show how use cases relate to each other and to actors across the system boundary, we distinguished this use of the diagram type by referring to them as analysis object diagrams. One analysis object diagram was created for each use case (see Figure 4-27 for an example), though on a second pass it was found closely related use cases could be supported by a single analysis object diagram.

The creation of the analysis object diagrams, threw up issues that were not expressed in the use case descriptions, and thus needed further consideration. For example, in the subscription management component the cardinality of the relationship between a customer subscription and a SLA was unclear. Once these issues were resolved the results were fed back into the use case descriptions to ensure there were no

inconsistencies with the analysis model. A consolidated analysis object diagram was also produced showing all interface objects, control objects, and entity objects that they manage, thus providing an overview of the analysis model.

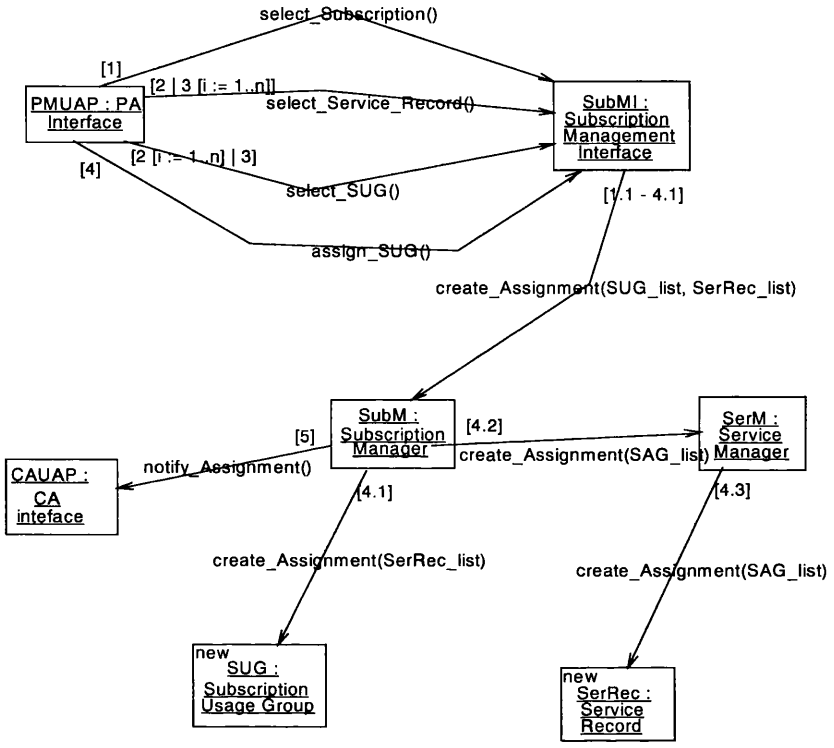


Figure 4-28: Example of a Collaboration Diagram for a Use Case

Collaboration and sequence diagrams were also used to show the dynamic behaviour of the analysis object in enacting a use case. Figure 4-28 gives an example of such a collaboration diagram.

The design model used was a straight-forward UML class diagram representation of the IDL interfaces and their parameter types with component diagrams used to represent the COs. There was a clear mapping between control objects in the analysis model and computational objects already defined in the subscription management component design. Boundary objects (e.g. Subscription Management) mapped to individual initialisation, query, and management interfaces in the design model, which in turn mapped to IDL interfaces. Entity objects were mapped to design object

representing IDL operation parameter structures. A smaller group of many to one relationships also exist. This occurred when distinct entity objects were identified from the use case descriptions but their characteristics were similar enough to justify a common, possibly abstract, design object. The analysis model failed to identify some objects and relationships present in the existing design. For example, the analysis model contained no Subscription Portfolio object, a collective holder for a number of customer subscriptions. Nor did it model the complex association between a Service Record and a Subscription Usage Group. These examples represent the designer's skill in being able to harness past experience to recognise suitable design patterns and take into account non-functional requirements such as performance issues, caching requirements, etc.

For publishing the facade, use cases were written in a word processor that could generate HTML output. The analysis and design models were generated on the Paradigm Plus CASE tool as UML diagrams with additional relationships added for links between specific UML classes in different diagrams. Paradigm Plus was able to generate HTML pages for specific diagrams and for each class modelled. Paradigm Plus uses an object-oriented database for storing its UML models and provides a simple scripting language for querying that database. These features were exploited in order to augment the HTML generated by Paradigm Plus with the additional links needed to be able to trace between the analysis and design models. The publication of the facade on the web is recounted in more detail in [lewis99b].

4.5.1.2 Open Business Process Modelling Approach

The attempt to provide a standardised architectural framework for analysing business requirements for SMS have, as described in Chapter 3, centred either around the definition of distinct business roles and the reference points that exist between them, as in TINA, or on the definition of a common business process model as in the TMF's TOM. These two inputs were therefore chosen as the basis for a model that enabled business process modelling to be applied to the multi-domain problems of

SMS development, but in a way would support the on-going standardisation of service management functions within these two bodies.

The approach taken was to map the TMF business model onto the TINA business model. Before examining such a mapping however, the core differences between the two models must be appreciated. Firstly, the TMF's TOM defines general business processes in existing service providers. These may be human based processes or automated ones. Part of the intention of the TOM is to identify and prioritise which processes they wish to automate, and therefore which inter-process interactions would benefit from industry agreements. The TINA model restricts itself only to reference points that will yield automated interfaces. Also, the TMF's TOM is concerned only with service and network management processes, while the TINA reference points additionally cover issues of service and network control. TINA also assumes its DPE (essentially CORBA) will be used to implement reference point interactions, while the TMF's TOM makes no assumptions about implementation technology (this is addressed in the TMF's Technology Integration Map as discussed in Section 2.3). Functionally, TINA management is aimed specifically at managing TINA services (multimedia, multiparty, multi-way, mobile) and network resources (connection oriented, broadband), while the TMF model is less specific, but is derived from the management of more contemporary services and networks, i.e. POTS, Frame Rely etc. TINA also specifically covers information services, while these have not yet influenced the TOM to a large extent. Finally, the TOM prioritises issues of process interaction and information flow between processes, while the TINA business model and reference points are focused on the development of detailed reference point specifications, based on other ODP-based TINA specifications, with little attention directed at business process information flows.

The approach taken in mapping the TOM to the TINA business model and reference points is to identify which TMF processes operate in which TINA Business Roles. Note that some TMF processes may be present in more than one TINA Business Role. An initial mapping of the TMF business processes onto TINA business roles is given in Figure 4-29

The principal assumptions behind this mapping are as follows:

- The TINA Retailer role is the one that embodies the TM Forum Customer Care Processes. If an organisation operating in the Retailer role has to communicate with another organisation playing the Connectivity Provider role or the 3pty Service Provider role, then these other organisations will also have to play the Retailer role so that any Customer Care process-related interactions are performed via the RtR business relationship.
- The TINA Retailer is not concerned with any Network and System Management Processes.
- The Connectivity Provider role is only concerned with Network and System Management Processes.
- The 3pty Service Provider is only concerned with Network and System Management Processes related to the provision of service content, i.e. Network Provisioning, Network Inventory Management and Network Data Management.
- The Broker will be effected by Sales, Order Handling and Rating and Discounting processes in other business roles, so these processes are mapped onto this role to indicate this. This is not intended to cover the application of these processes to the Broker's own services (i.e. broker services), which should be addressed by an organisation in the Broker role also taking on the Retailer role.

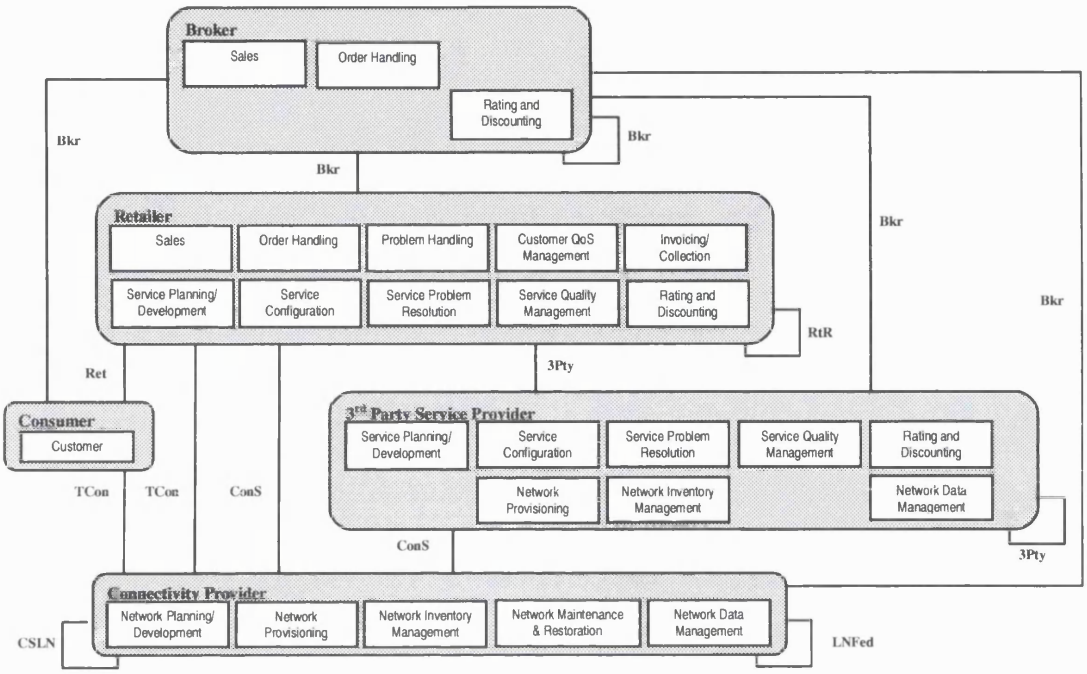


Figure 4-29: Mapping of TMF Business Processes onto TINA Business Roles

The TOM provides a model of suitable business processes which we are fairly confident reflects the typical operations of a service provider. This mapping, therefore, helps in the analysis of a Service Provider's business processes in order to identify where existing solutions, possibly available as reusable components implementing reference point segments, can be applied. The analysis of business processes is typically performed by identifying discrete activities and the events that propagate the control of execution of a task between activities. As described in Section 3.1.2, a common representation of such a control flow is event-driven process chains, and the inclusion of activity diagrams allows UML to support a similar type of model. The analysis of a management task in a specific business scenario can be initially described in terms of a use case giving the interactions of the system with the human roles involved in the task. These use cases can then be broken down into internal activities performed with the system, using a UML activity diagram. The activities can be placed within swim-lanes representing TMF business processes, possibly residing in different administrative domains. This will ease the identification of which existing TMF business agreements match the

requirements of the task at hand. The mapping of the TOM process onto the TINA business roles will then enable the identification of where TINA reference point definitions could apply. The following section outlines an example of the application of this business modelling technique from the FlowThru project.

4.5.1.3 Application of Open Business Modelling

The problem addressed by this example is the integration of management functions in the service and network management layers related to the fulfilment of customer orders for a switched ATM service. It was assumed that customers were not restricted to single terminal domestic users but would also comprise of corporate users with switched ATM customer premises equipment supporting a number of users via a single UNI.

In terms of the TMF's business process model, the problem domain encompassed the fulfilment process area, which consists of the following business processes:

- Customer Care Process: This involved the management of the interactions between the customer and the provider's management system, in particular translating customer requests and queries into appropriate system operations.
- Order Handling: This involved the receipt of orders for ATM services from a customer and the translation of these orders, after appropriate financial checks, into requests for network configuration changes needed to satisfy the customers order. It then involved tracking the progress of these orders to completion.
- Service Configuration: This involved the installation and configuration of a service for a specific customer, including any customer premises equipment.
- Service Planning and Development: This involved the design of the service capabilities to meet market requirements for services within required costs and exhibiting required manageability characteristics.
- Network Provisioning: This involved the reconfiguration of network resources so that network capacity is ready for the provisioning of services.

- Network Planning and Development: This involved designing the network capability to meet specified service needs at the desired cost and within operational constraints, determined principally by service level agreement with customers.
- Network Inventory Management: This involved the installation of physical equipment in the network.

Within the context of TINA, the problem may be expressed in terms of business roles and reference points between these roles. The business roles identified for this scenario were the Consumer role taken by the ATM service customer organisation while the ATM service provider organisation took the Retailer and Connectivity Provider business roles. Based on the business model from the previous section, Figure 4-30 shows the overall business context of the problem being addressed in terms of the business roles played by the organisations involved, the business processes undertaken by those roles and the TINA reference points that exist between those roles.

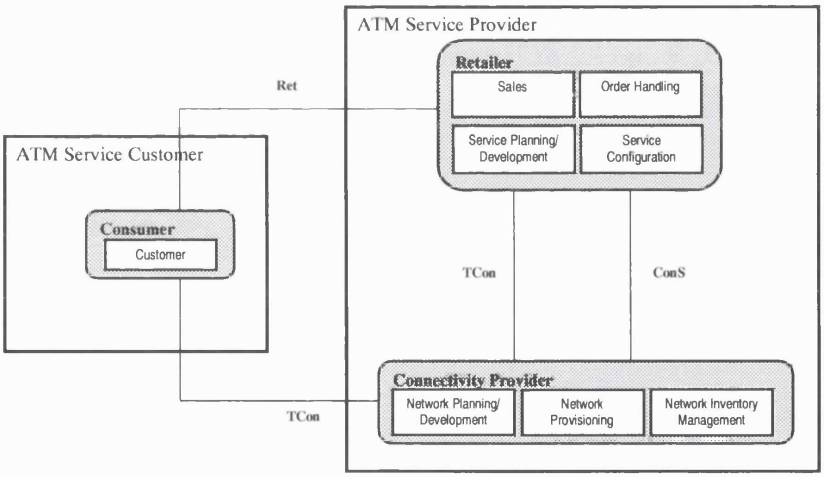


Figure 4-30: Business Process to Business Role Mapping for ATM Service Fulfilment

As the business problem under examination did not involve multiple providers the need for TINA federation reference points for the roles within the ATM Service Provider were dropped. The remaining relevant TINA reference points were:

- Ret reference point: This was concerned with access control between the Consumer and the Retailer, the discovery and commencement of operational and management service offerings and the control and management of session, stream flow bindings and stream flow content.
- ConS: The control and management of connections between network access points.
- TCon: The negotiation of and control of network level interconnection, which is technology specific.

As the focus of this work was on the development of functions in what was typically described as the management plane, aspects of network and service control were assumed but not analysed or developed any further. Therefore the TCon reference point was not examined, with the assumption that existing UNI protocols would be used for connection set-up. Instead the focus was placed on the management interactions that occurred across the Ret and ConS reference points. Work on these reference points within TINA-C had mostly been concerned with the control of user session. This provided further contextual information on the service to be managed. The Ret reference point was derived form the TINA Service Architecture and included interfaces to the subscription management component use in the previous case studies. The ConS reference point was based around the TINA Network Resource Information Model (NRIM), however this had been enhanced with additional network configuration, planning and restoration management features by the REFORM project as described in [georgatatsos].

These existing TINA-based models provided a core set of concepts that could be used in further analysing the requirements of the ATM service fulfilment business problem and the interactions between the business processes identified as being relevant to this problem. These models had been captured in the facades that had

been generated for each of the components brought to the problem. These facades yielded key information definitions, such as Network Access Point, Class of Service and Service Usage Group, represented by entity objects in their analysis model

To analyse the problem further, business roles within the business stakeholders were analysed to determine the responsibilities they owed to each other. The set of responsibility relationships was represented as a UML class diagram in Figure 4-31.

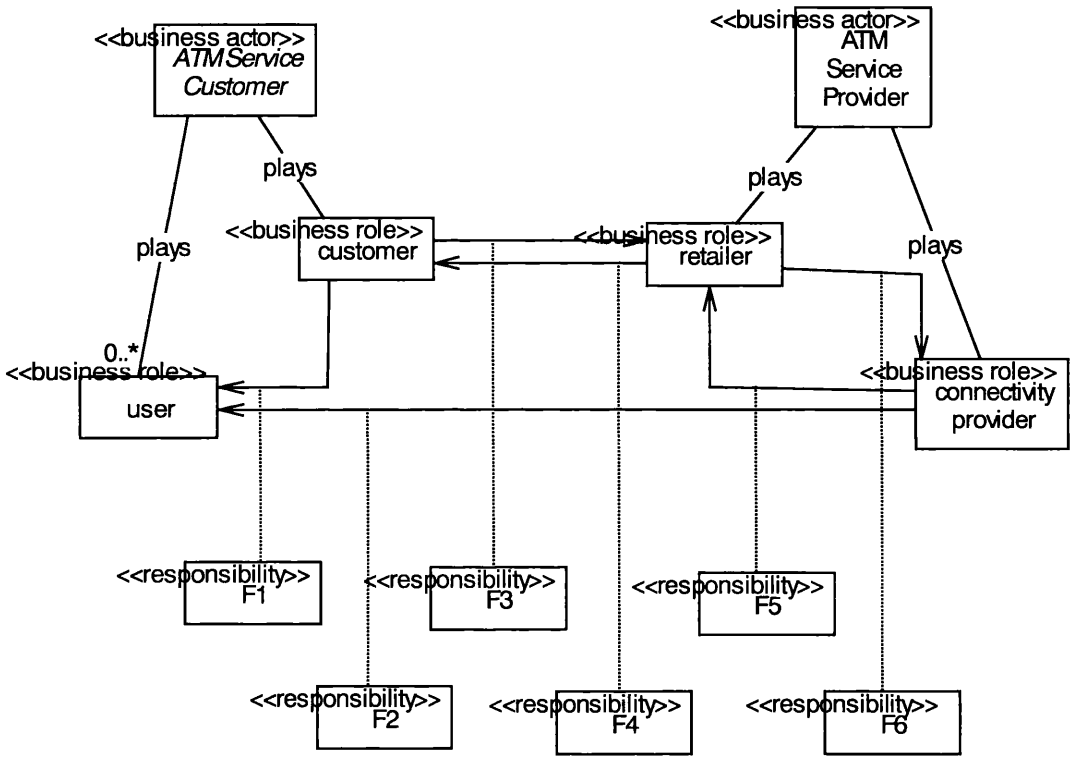


Figure 4-31: UML Class Diagram of Responsibilities Between Business Roles

The responsibilities from Figure 4-31 were defined as text descriptions. Based on these responsibilities use cases were drawn up documenting the functionality of the whole business system from the point of view of actors whom play the part of the business roles. These use cases cover the following functionality and are summarised in the UML use case diagram shown in Figure 4-32.

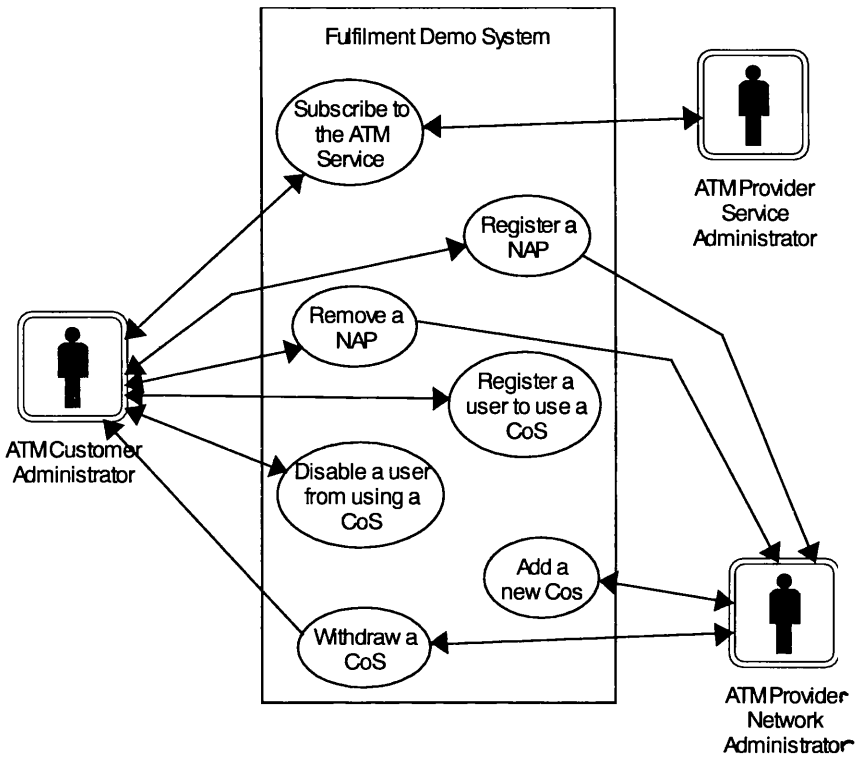


Figure 4-32: Use case Diagram for ATM Service Fulfilment Scenario

These use cases were then analysed to refine the relationships between objects representing the major entities that appeared in the use cases and were related to entity objects in the component facade's analysis models. These relationships were summarised in a UML class diagram shown in Figure 4-33.

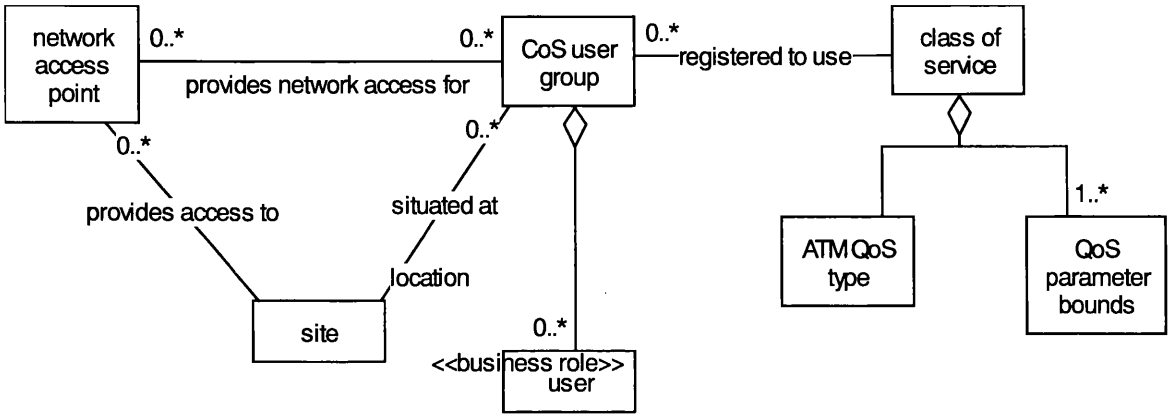


Figure 4-33: Relationships Between the Main Entity Objects Identified in the Use Cases

The use case descriptions did not however yield much direct analysis of the internal business processes that were required within the service provider's domain, and it was here that the required interactions between service and network management components needed to be identified and resolved. It was at this point, therefore, that individual use cases were broken down into separate activities modelled in a UML activity diagram. These activities were grouped into swim-lanes according to the TM Forum business process into which they best fit. The activity diagram for the "Subscribe to ATM Service" use case is presented in Figure 4-34.

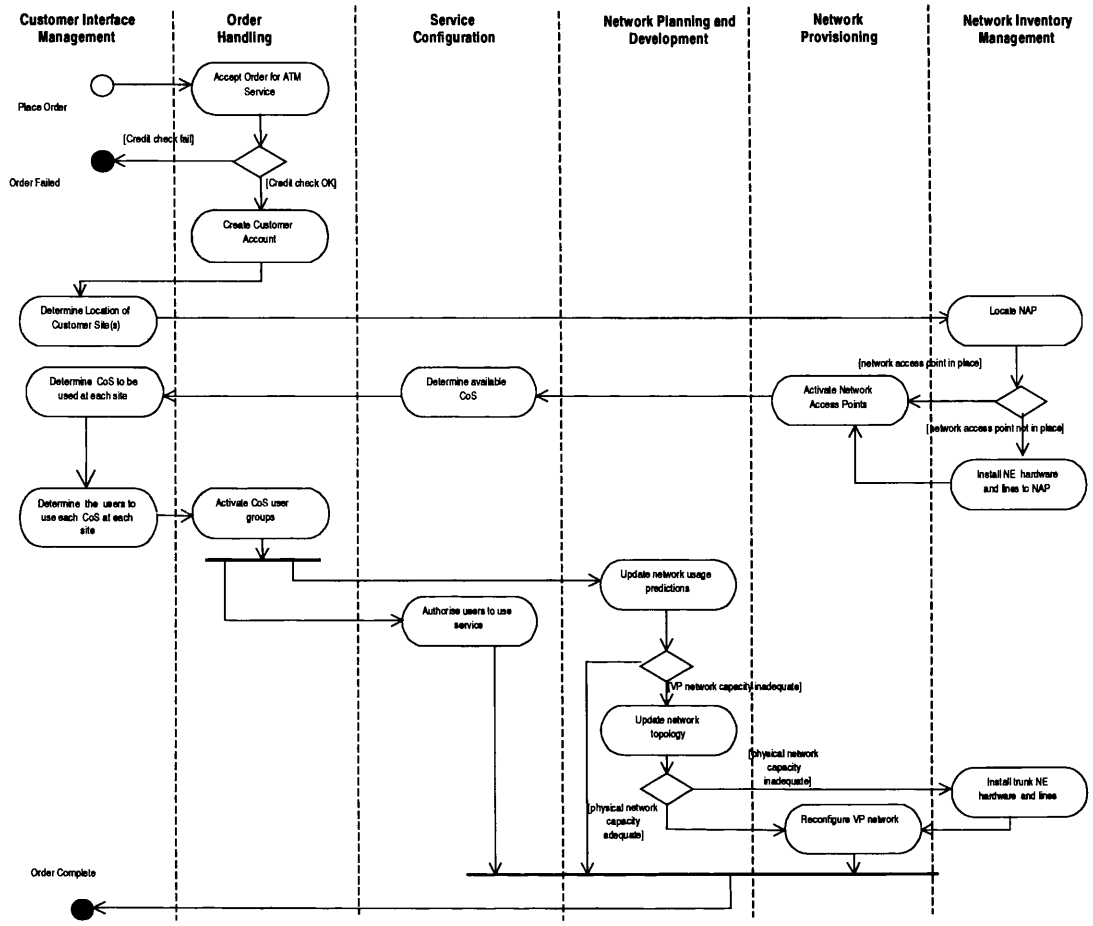


Figure 4-34: UML Activity Diagram for the Subscribe to ATM Service Use Case

By comparing these activities to the use cases in the imported components' facades, a mapping could be obtained of which activities could be handled by which components. From this a clearer picture of the interactions required between the component was formed. This enabled the identification of specific requirements for modifications to components in order that they satisfy the overall system requirements. For instance, the activities in the Order Handling swim lane in Figure 4-34 and their interactions with the Customer Interface Management swim-lane activities were identified as ones that could be performed by the subscription management component. This analysis indicated, however, that the design of this component needed to be modified to support asynchronous interactions with components performing activities in the Network Provisioning and Network

Planning and Development swim-lanes, as these could result in considerable delays. The TMF already had an interface agreement, Service Provide to Service Provider Order Handling [nmf-504], that addressed aspects of this process area. This standard contained abstractions for tracking orders in slow response situations, which therefore could be applied to the modification of the subscription management component.

The overall system analysis model was defined in terms of the interactions between analysis model elements from the imported component facades, and in particular the bindings between their boundary objects. The overall system design model consisted of the modified IDL for each component and detailed sequence diagrams showing inter-component interactions that enacted the system's use cases.

4.5.2 Evaluation and Results

The experiences of this case study relate the generation of the facade for a pre-existing component and how this may be integrated into the development of an SMS in a way that ensures the satisfaction of business process requirements and ready integration with existing open standards. The facade presents the component model at levels of abstraction, i.e. as use case and analysis models, that facilitate the components integration into the SMS. It was found that the division in the facade between the analysis model and the design model provided a good basis for delineating between the exposure of internal details of a component needed for its features and capabilities to be understood. At the same time the facade hid all detailed design issues except those relating to the interfaces via which it is re-used.

Publishing the facade in HTML with a structure suitable for re-users to make best use of the traces between elements in the different models was found to be relatively simple with Paradigm Plus. The fact that the same teams were involved in facade generation and the design of the systems that reused the components meant however that the effectiveness of the facade in component selection and comprehension could not be assessed very objectively.

The business process and reference point mapping model was found useful in bringing together the business process model approach to establishing management requirements from the TMF and the component-based reference points defined in TINA. This assisted the understanding of developers with backgrounds in TINA who needed to use their systems to satisfy business process requirements. This mapping has been presented to the TMF and TINA-C by the author. It has been received with interest by both and at the time of writing is forming the basis of negotiation on a possible liaison agreement between the two bodies. Such a mapping, combined with the representation of reference points as the facades of their constituent components, points the way to the integration of existing component and interface specifications with business process driven SMS development. It also provides a path to expanding the scope of reference points based on existing business process analyses.

The above observations are based on the author's own modelling experience with the teams developing the subscription management component facade and the ATM fulfilment business scenario. In addition, however, a questionnaire was used to get a more structured view of the developers' experiences, against which the above observations may be compared. The questionnaire was structured in a very similar manner to the one in Case Study 4 and the results analysed in a similar manner. As well an initial and a final section similar in structure and purpose to the corresponding ones in Case Study 4 questionnaire, the following sections were present:

- Section two was to be answered by those involved in establishing the business process requirements for the systems being developed. It addressed the usefulness of the business process modelling elements in analysing business process requirements (Q2.2).
- Section three was to be answered by those involved in facade generation. It addressed the usefulness of use case modelling elements when developing the analysis (Q3.2) and design models (Q3.3) and of the analysis modelling elements when developing the design models (Q3.4).

- Section four was to be answered by those involved in the design of a trial business system. It addressed the usefulness of the business process modelling constructs (Q4.2), facade modelling constructs (Q4.3) and general system modelling constructs (Q4.8) in designing the systems. It also addressed the usefulness of the facade constructs in the modification of the components for reuse (Q4.6).

In addition to the above questions some broad questions on the overall effectiveness of the techniques were posed with space to provide more free flowing responses. The questionnaire did not address the frequency and types of problems encountered by developers as in Case Study 4. Also, the timing of the questionnaire mean that impact of the modelling techniques used on the implementation and testing/integration stages of development could not be addressed.

Fourteen completed questionnaires were returned and are presented in Appendix 1 (Section 8.2) in the same manner as for the Case Study 4 questionnaire responses. The responses mostly were in line with the author's observations above and the findings of the previous case studies. The business process modelling was rated highly for both system and component development. The facades were also rated highly by system designers, though the rating for the facade's use case model, indicates that this was not clearly related to the analysis and design model in all cases. It is notable that the usefulness of the two CASE tools used, Rational Rose and Paradigm Plus, was much higher than for their use in Case Study 4, possibly indicating better developer familiarity with these tools. For the system designers, being able to work at the level was seen as very useful.

The responses to the broad questions were as follows:

- In response to the question "Do you think the use of component facades resulted in a better-designed component?", six replied yes, three no and four did not express an opinion. Comments were made that in most cases the component design was fairly mature, so the facade generation was purely a documentation exercise.

- In response to the question "Do you think the design of the trial business systems was made easier or not by the use of the component facade structure?", seven replied yes, two no and five did not express an opinion.
- In response to the question "Do you think the modification of components for reuse in the trial business system was made easier or not by the use of facades?", five replied yes, none no and nine did not express an opinion.

In response to a further open question on what areas where not addressed by the methodological guidelines used but that could have been useful, comments were made on:

- The lack of common guidelines for the structuring of IDL module, type and interface definitions into appropriate groups.
- The lack of guidelines for expressing the exchange of asynchronous events between components.
- The difficulties in describing fully the reasons why components have the functional scope that they do, possibly requiring a more structured conceptual framework.
- The lack of working traceability between models under development.
- The general lack of clarity in the guidelines themselves, many found them difficult to follow.

5. Results and Synthesis

This chapter uses the results from the state of the art analyses in Chapter 3 and the case studies in Chapter 4 to synthesise a set of recommendations for an open SMS Development Framework. Based on these recommendations, a development framework is specified consisting of specific methodological guidelines and a discussion of suitable architectural guidelines. The development framework is then assessed with reference to the goals laid out for such a framework in Chapter 2.

5.1 General Recommendations

Case Studies 1 and 2 provided evidence that a scenario led development approach was well suited to the development of TMN systems, especially when several development teams were working in parallel. The primary benefit was found to be the way in which scenarios could be used to ensure the sometimes divergent work of separate teams was kept focussed on the original requirements of the system. This was shown to be the case both when SMS for different organisational domains were developed in parallel and when subsystems used within domains were developed in parallel. Case Studies 4 and 5 introduced Jacobsen's concept of a use case, which, though similar in aim to the scenarios used in Case Studies 1 and 2, are better defined structurally and more widely accepted in industry. Evidence from the questionnaire conducted for Case Studies 4 and 5 shows that use cases were regarded as one of the most useful techniques used in these case studies. The results from the questionnaire for Case Study 4 and 5 show that use cases were useful for both system development and component development. It can therefore be recommended that:

Recommendation 1: *Use cases should be adopted as the basis for mechanisms for capturing functional requirements for SMS, COTS management software and interface standards. They should be used, together with corresponding dynamic modelling of related business and object models, as a mechanism for ensuring that later development activities stay focussed on requirements.*

Though use cases were found useful for analysing the functional requirements imposed on an SMS or a reusable component by its environment, it did not always provide a clear route back to business driven requirements. Case Study 2 introduced the technique of role-based analysis, which was used to express the business relationships that existed between different organisational domains by describing the roles they played towards each other and the responsibilities that these roles exhibited towards one another. Case Study 3 showed that techniques for refining responsibility descriptions were not very usable, so role-based analysis is recommended for refinement only to the level of the responsibility descriptions, where they are then supplemented by use cases. Role descriptions, however, only focus on the business interactions between organisations and between human actors within organisations. Case Study 5 introduced the popular technique of business process analysis, inspired by its application in the TMF's Telecoms Operations Map. This technique allowed specific tasks to be broken down into activities within an organisational domain, driven by the role-based and use case-based analysis of the external behaviour of the domain. Case Study 5 also showed the practicability of combining business role analysis and business process analysis in applying existing solutions of SMS development. A combined functional architecture based on the mapping between TMF business processes and TINA business roles has been suggested by the author as practical mechanism for aligning the work of the two bodies in the management area. With this mapping the TMF may benefit from TINA's component-based, segmented reference point approach to standardisation while TINA may benefit from a wider set of business process driven requirements. This mapping is currently the basis of on-going liaison work between TINA-C's Service Management working group and the TMF.

Recommendation 2: *The capture and initial analysis of requirements for SMS should be based on a combination of use cases, role analysis and business process analysis. A functional architecture based on the mapping of business processes onto business roles should form the basis of a functional architecture for the ongoing standardisation of service management interfaces.*

The case studies have applied a variety of textual and graphical notations to the development of SMS, reusable components and potential interface standards. Case Study 1 attempted to use notations recommended in [m3020-95] but, though GDMO is important for defining interfaces to CMIP-based systems, the definition of analysis level notations was not well supported. Case Study 3 applied ODP viewpoint notations as recommended in TINA's development guidelines. Though these were individually useful, the close coupling between the informational and computational viewpoints and the lack of tool support for maintaining the mappings between the model elements for these viewpoints made this approach impractical. Case Studies 4 and 5 showed that the different modelling diagrams offered by UML were useful in the development of SMS and reusable components, particularly in the light of the widespread tool support for this notation. This experience revealed that UML possesses some shortcomings, amongst others, in the dynamic modelling of multi-interface objects, but that its stereotyping mechanism enabled many of these to be circumvented. It was also necessary to use textual interface definition languages such as IDL or GDMO to exploit existing distributed computing platforms, and the round trip integration of UML with such languages needs to be better understood and common agreements reaches. However, the standardised status of UML together with its increasingly widespread use by developers of all persuasions, spurred by widespread tool support leads to the following:

Recommendation 3: *UML should be used as the common notation for the exchange of models between all SMS development stakeholders, supplemented by machine processable interface definition languages in the design models where necessary.*

A further problem encountered in the case studies was in agreeing a common level of abstraction for the exchange of models during the analysis and design stages. In Case Studies 2, 3 and 4 detailed analysis and design modelling was performed with a combination of class diagrams, notations showing discrete functional units and their interfaces, sequence diagrams and interface definition languages. However, the generality of these notations made it too easy for developers to blur the distinction between analysis and design models, reducing the usefulness of being able to work at

different levels of abstraction. Jacobsen's analysis types, as introduced in Case Study 5, were found to help in making the distinction between analysis and design models in a way that was simple to understand and easy to apply. Hence:

Recommendation 4: *Jacobsen's analysis object types should be used as the basis for describing analysis models, thus making them distinct from design models.*

The experience from Case Studies 3 and 4 in using models from a variety of different sources illustrated the difficulty in working in such an open manner. To be able to use a model from any one of these sources a large investment had to be made in understanding the architectural framework in which the model resided. This formed a barrier to reuse of open solutions from different sources and encouraged tie-in to a single source of models. Case Study 5 addressed this problem by introducing the facade modelling construct which packaged together the requirements, analysis and design models for a particular solution so that it can be reused with relatively little reference to an encompassing architectural framework. Case Study 5 provided an indication that the facade construct was both useable and useful in combining solutions from different sources. These solutions represented reusable components and potentially open interface definitions. Hence:

Recommendation 5: *The facade modelling construct, packaging requirements statements, analysis models and design models should be used for presenting SMS solutions for reuse, both for reusable components and for interface standards.*

These recommendations are not individually remarkable as similar ones can be found in many software engineering texts. What they do represent, however, is a set of recommendations that have been tested and validated in the context of SMS development. They can therefore be used to form the basis of methodological guidelines for an open SMS development framework with a higher degree of confidence than the many other methodological techniques that could be applied in this context but remain untested. The next section synthesises such an open development framework.

5.2 Synthesis of Open SMS Development Framework

This section synthesises a development framework based on the analysis of SMS development frameworks in Chapter 3, the experiences from the case studies presented in Chapter 4 and the resulting recommendations presented in the previous section. Consistent with the thesis statement and the results of the state of the art analysis, the development framework attempts to be prescriptive only in its methodology guidelines, which are presented in the next section, while providing only loose architectural guidelines in the subsequent section.

5.2.1 Methodological Guidelines

The methodological guidelines presented here are structured according to the analysis given in Section 3.5. They are focused on the relationships represented in the generic SMS developer stakeholder model of Section 2.1 as this is where they are expected to yield the widest benefit as part of a common development framework used by all stakeholder types. The guidelines are expressed in terms of the notations and meta-model to be used, and in terms of how these should be applied to the development processes for each development stakeholder type, based on the individual process model identified in Section 3.5. The guidelines are restricted in scope to the issues covered by the recommendations given in the previous section.

5.2.1.1 Notations and Meta-model Definition

This section defines the notations that should be used by SMS development stakeholders and the meta-models, i.e. the structure of information, to which models expressed in these notations should conform. Based on general Recommendation 3, the core notation used is UML, specifically the OMG's current version 1.1 [ad/97-08-01]. UML is, however, a general purpose modelling language and its designers acknowledge that it is necessary to extend and profile it to suit software development requirements of specific problem domains. This section therefore uses the UML stereotyping mechanism to propose extensions to UML for the SMS development framework. This is presented in terms of stereotypes defining new modelling

elements and the meta-model that defines the relationships of these elements with each other and with existing UML v1.1 elements. Class diagrams are used to show the relationships between these stereotypes and their relationships with existing UML model elements, which are identified for convenience by the stereotype marker <<uml1>>. UML model elements are written in double inverted commas when first introduced, and subsequently where needed to avoid ambiguity. The specific modelling constructs defined here are:

- A Use Case Model
- A Business Requirements Model combining Business Process, Business System and Use Case Models
- A Facade modelling construct defined here as a Projection as explained below.

UML already defines a "facade" stereotype of the "package" model element. A facade is defined as the public view of the content of another package, containing only elements from that package and none of its own. This definition of a facade conflicts with the usage of the concept intended by Recommendation 5. This, supported by the experience from Case Study 5, intends the facade to be a public view of an internal model, but conforming to a specific structure. The condition that the public package is not necessarily a strict subset of the internal package differentiates the modelling construct defined here from the UML facade. For this reason the alternative stereotype designation of a Projection is defined.

5.2.1.1.1 Use Case Model

UML specifies a stereotype of the model element "model" called "Use Case Model", which defines the functional requirements for a system. It contains only "use cases" and "actors", the "interaction" relationships between them, the "generalisation" relationships between actors and the "extends" and "uses" relationships between use cases. This definition and the supporting "use case diagram" is necessary to meet the requirements of the SMS development framework, but is not sufficient as it does not define the structure of use case definitions. Several different conventions for use case

descriptions have been proposed, but these guidelines define the following simple convention to ensure commonality in how use cases are expressed.

A use case description is a textual entity that uses natural language. It should consist of at least the following elements:

- A name that uniquely identifies the use case description. This requires a distinguished naming scheme that identifies the system the use case is being used to describe and the organisational unit conducting the analysis. The name should be meaningful and express the task being addressed by the use case.
- A primary actor from the set of actors that defines the system's environment. Each use case should have one and only one primary actor, which is the one drawing primary benefit from the execution of the use case.
- A set of preconditions that describes the state of the system immediately prior to the use case being enacted. Each precondition should be uniquely identifiable within the use case description. A precondition may consist of a reference to the post condition of another use case from the same use case model.
- A description of the use case in terms of a sequence of steps describing the operations performed by the use case in the order in which they occur. A step may include a condition, which must be satisfied in order for it to be conducted. A step may refer to another use case in the same use case model, this reflecting a "uses" or "extends" relationship.
- A set of post-conditions that describe the state of the system immediately after the use case has been enacted. Each post-condition should be uniquely identifiable within the use case description.

5.2.1.1.2 Business Requirements Model

The Business Requirements Model is a stereotype of "model" that aims to support the identification of requirements in complex multi-domain situations. It consists of a Business System Model together with a Use Case Model, of the type described in the previous section, and a Business Process Model. All three are UML "model"

stereotypes. Model elements in the Business System Model are associated with model elements from the other two models as depicted in Figure 5-1.

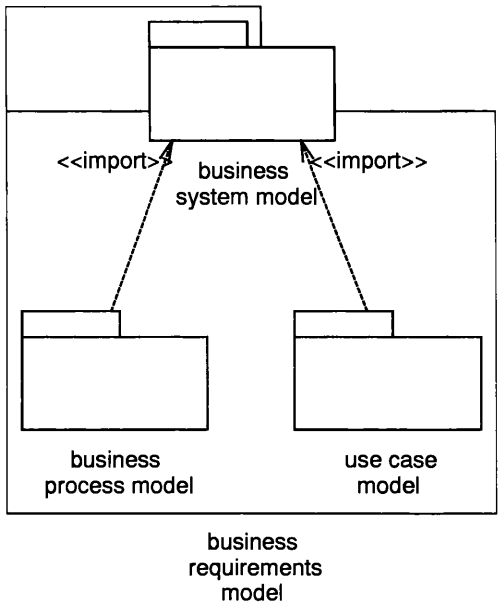


Figure 5-1: Structure of the Business Requirements Model

The contents of the Business System Model and the Business Process Model, and the association between elements in all three models are summarised as follows:

Business Process Model:

This contains the following modelling elements::

- *Business Process:* This represents a process that must be conducted to perform the business functions required of the system. It is a high level identification of an ongoing business task rather than specific identification of an activity with defined initiation and termination conditions and the flow of control between them as used in UML activity diagrams.
- *User:* This acts as a source and/or a sink of information that must be handled by one or more Business Processes. The set of users in the model defines the environment that motivates the flow of information between business processes. A User must be mapped to an actor in the use case model.

- *Information Flows:* This represents the flow of information that may pass between Business Processes or between Business Processes and Users.

Business System Model:

This contains the following modelling elements:

- *Organisational Domains:* This represents an organisation involved in the business scenario under analysis, e.g. a service provider or a customer.
- *Business Role:* This is a role played by a User within a specific Organisational Domain, e.g. service user or service administrator.
- *Responsibility:* This is a unidirectional relation between two Business Roles defining the contractual obligation one has to the other, e.g. "pay charges by due date".
- *Service Management Systems:* This represents the system under analysis, which may be one of several operating within an Organisational Domain.
- *Contract:* This represents the set of functions that may exist between two Service Management Systems.

The following relationships exist between the modelling elements in a Business Requirements Model. They are also depicted in Figure 5-2:

- Business Roles should map one to one to actors in the use case model, so the descriptions of a Responsibility between two Business Roles should be consistent with the corresponding actor to use case interactions and User to Process Information Flows.
- Business Processes should be wholly instantiated within an Organisational domain.
- Individual Systems should exist wholly within one Organisational Domain

The identification of these modelling elements and their relationships enable the business requirements to be expressed in terms of requirements upon specific contracts in terms of Responsibilities and Information Flows. This is particularly

useful for defining reference points between Organisational Domains that do not involve direct interactions with actors and are therefore not addressed directly by the use case model.

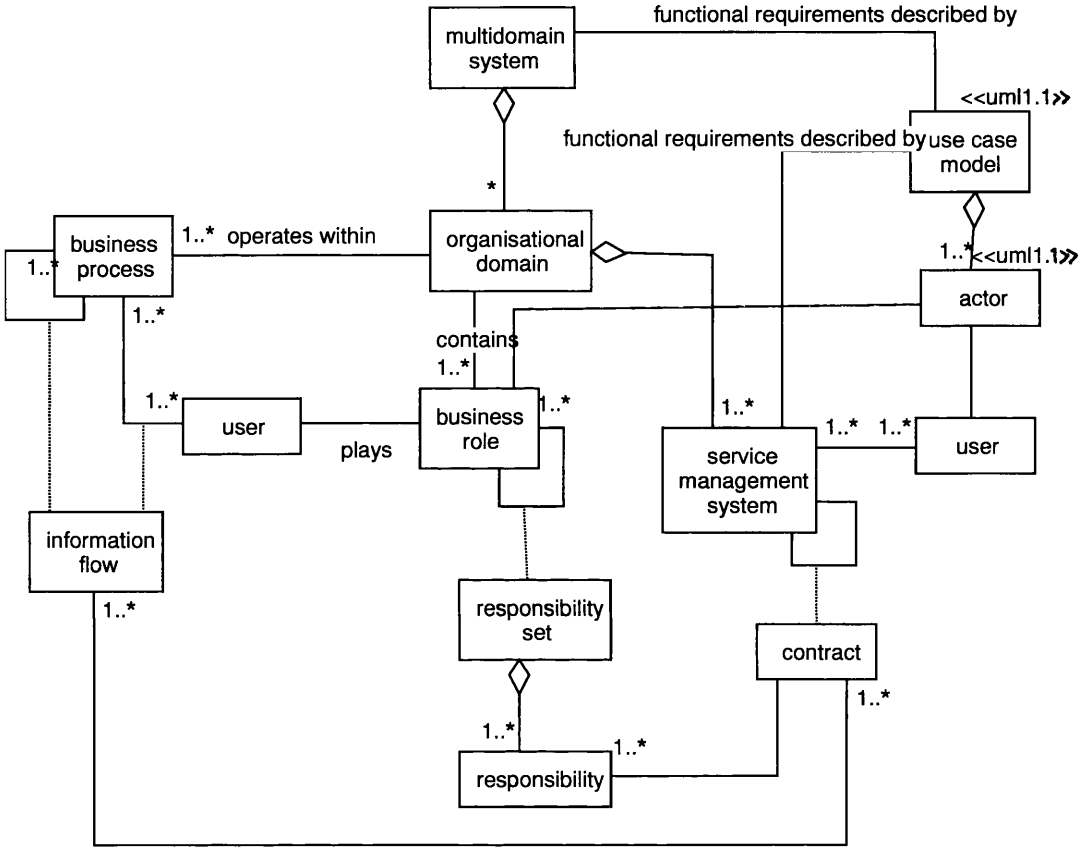


Figure 5-2: Relationships between the Elements of the Business Requirements Model

The process of generating a Business Requirements Model consists of the following steps:

First, establish a multi-domain organisational model (part of the Business System Model) that identifies Organisational Domain, Business Roles with those domains and Responsibilities between them. This can be done using a UML class diagram together with corresponding textual Responsibility descriptions as in Case Study 5 (see Figure 4-31).

Second, establish a use case model where the actors represent the different Business Roles from the multi-domain organisational model and the use cases are described at

the multi-domain level. An example of this is the fulfilment trial business system use case description presented in Case Study 5 (see use case diagram in Figure 4-32).

Third, establish a Business Process Model where the users are the different multi-domain user case actors. This can be done using a component diagram to show which Business Processes interact with which Users in which Organisational Domain. Figure 5-3 shows such a diagram for the processes, roles and domains identified in Figure 4-30 for the fulfilment trial business system described in Case Study 5.

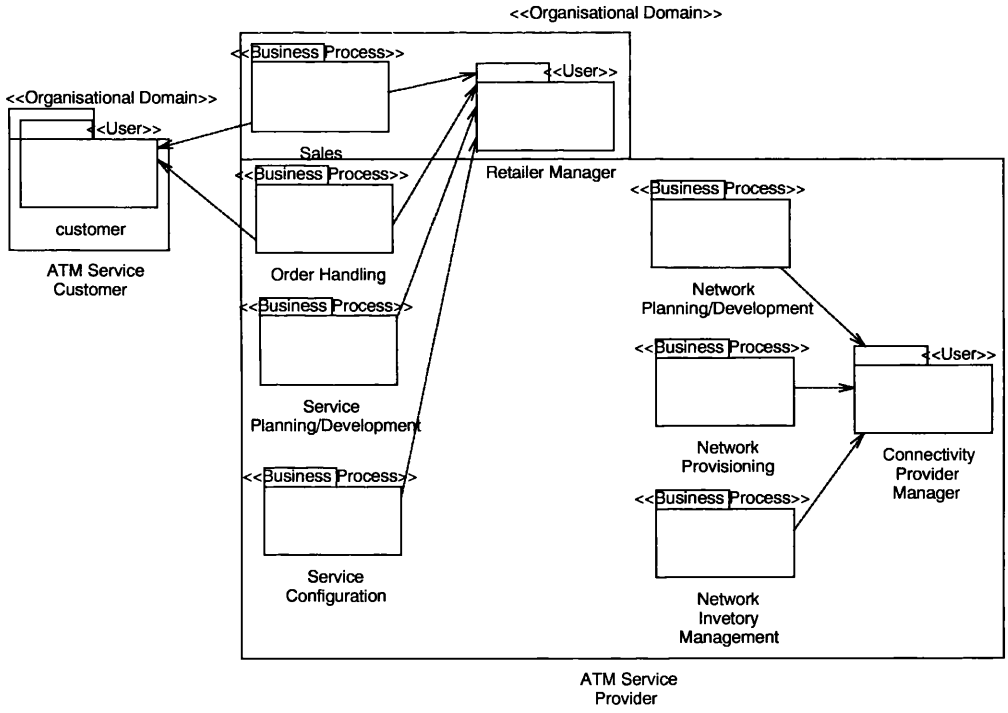


Figure 5-3: Example of static Business Process Model using a UML Component Diagram

Fourth, refine the Business Process Model to show for each multi-domain use case the information flow that must flow between Business Processes and between Business Processes and Users. This can be performed using UML sequence diagrams. An example is given in Figure 5-4 for the processes and users in Figure 5-3, for the "subscribe to ATM service" use case from the fulfilment trial business system presented in Case Study 5.

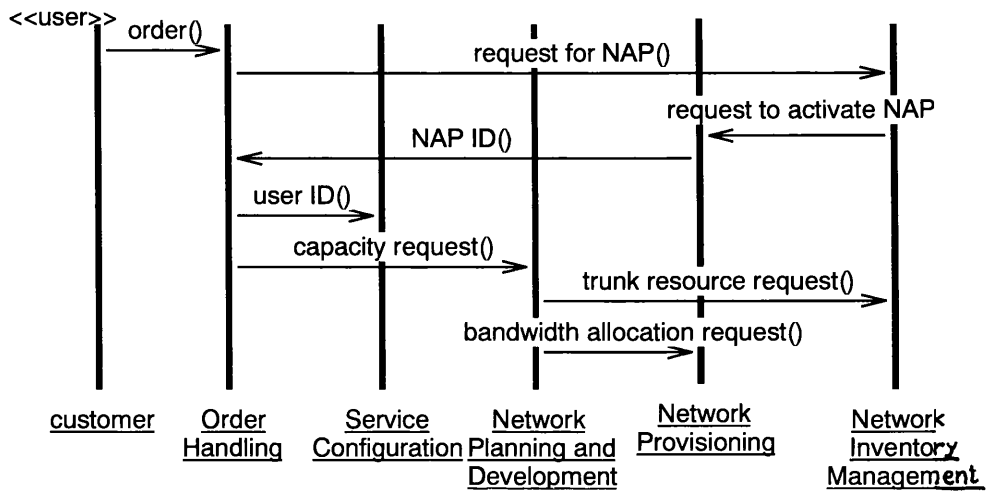


Figure 5-4: Example of Dynamic Business Process Model using a UML Sequence Diagram

Fifth, establish a Business System Model that shows the SMS under analysis and the other SMS and the Business Roles with which it interacts in the same or in collaborating Organisational Domains. The model should identify the Contract via which interactions are performed. This model can be represented using a UML component diagram, an example of which is give in Figure 5-5 for the SMS making up the fulfilment trial business system.

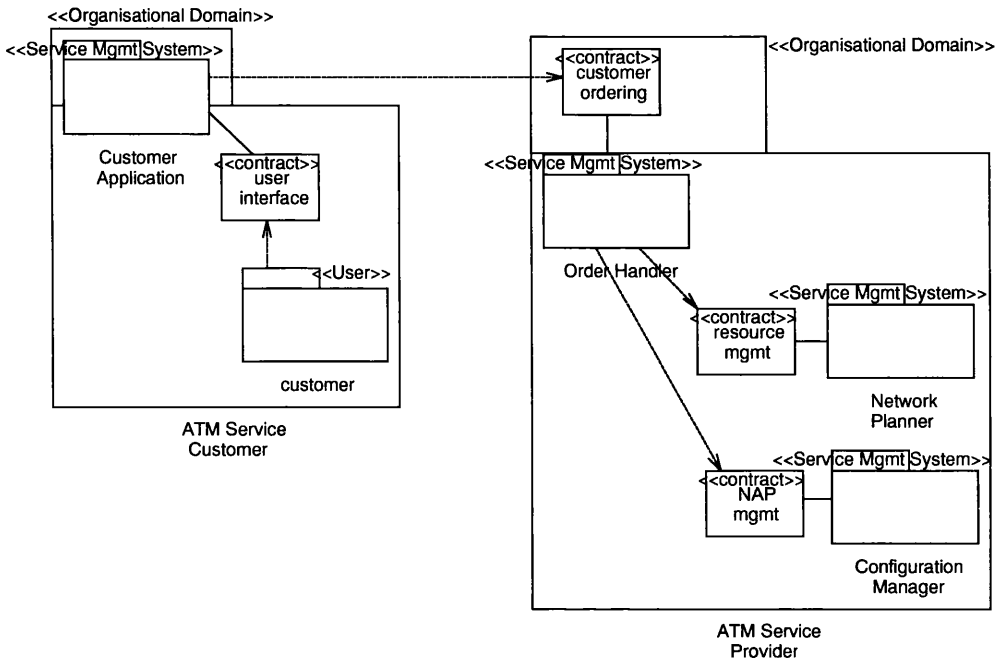


Figure 5-5: Example of SMS Level Business System Model using a UML Component Diagram

Finally a use case model can be generated for the SMS under analysis, with actors representing the users and other SMS with which it interacts. The individual use cases involved should be triggered by inward Information Flow for a Business Process handled by this SMS, as identified by the dynamic Business Process Model. As the actors represent the Users and SMS that interact with the SMS under analysis via Contracts, the aggregation of the all interactions between the user cases and a specific actor will define the functions required at the corresponding Contract.

5.2.1.1.3 The Projection Modelling Construct

A Projection allows models to be exchanged between development stakeholders whose internal models may not yet conform to the common structure used in the Projection. In such cases a mapping must exist between the meta-model used internally by the stakeholder and the Projection meta-model. Case Study 5 already demonstrates that such a mapping can be defined between the model elements of a facade and ODP-based TINA models, and the same applied to Projections. As with

facades, a Projection can provide a selective view of a system, revealing only the details judged by the owner of the system as needed for a specific type of user of the system, e.g. a software reuser or a specification reuser. Consequently a system may support several Projections in parallel.

The Projection construct has a more defined structure than the existing UML facade. This structure is shown in Figure 5-6.

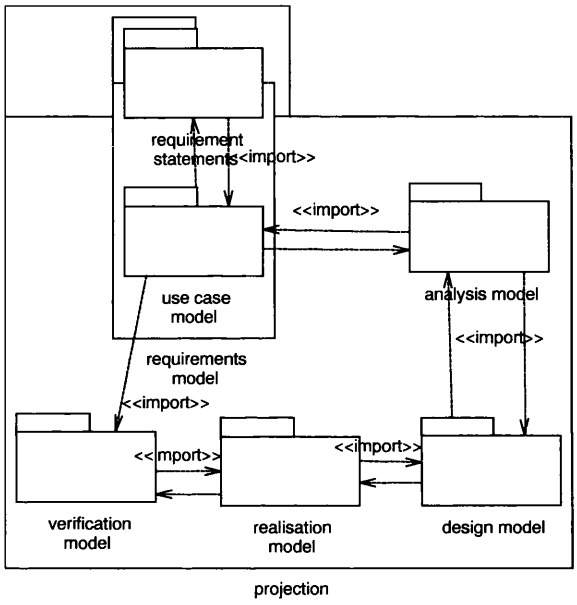


Figure 5-6: Structure of the Projection Modelling Construct

A Projection consists of the following elements, each a stereotype of "model", and dependencies between their modelling elements:

Requirements Model:

This contains a complete requirements statement for the system concerns addressed by the Projection. It consists of two parts:

1. A set of textual requirements statements that are uniquely identifiable within the context of the Projection and which fall into one of the five requirements categories defined in the TMF development methodology, i.e. Structural Information, Dynamic Information, Abnormal Conditions, Expectations and Non Functional Requirements and System Administration Requirements.

2. A use case model of the system being modelled addressing only the concerns relevant to the details revealed by the Projection.

Analysis Model:

This provides an analysis of the requirements presented in the Requirements Model. It contains: classes of the analysis object types defined by Jacobsen, i.e. the control, boundary and entity object types; actors that place requirements on the system and identification of interactions between analysis objects and between analysis objects and actors. Interactions may be classified by the one or more of the following types: create, delete, read, and modify. The static view of the Analysis Model may be represented in an analysis object diagram, which is a class diagram that supports the analysis object stereotypes. In parallel the Analysis Model should contain collaboration diagrams which show the dynamic behaviour of the classes in the analysis object diagram in terms of messages that pass between instances of them. The structure of the Analysis Model is driven by that of the Requirements Model. Each use case in the latter should be reflected by at least one analysis object diagram and one analysis collaboration in the former. In addition the analysis actors should have a one to one mapping to the use case model actors in the «requirements model». These model elements and their relationships to each other and to elements in the use case model are depicted in Figure 5-7.

Design Model:

This defines a view of the design details of the system judged sufficient by the system designer to allow the use of the system by others. Typically this will consist of:

- A description of functional structure of the system in terms of components and the interfaces they offer to and require of external entities and optionally each other. This may be expressed in terms of a component diagram.
- A definition of the interfaces offered to, and required of, external entities defining interface operations, their parameters and exceptions. This may be

expressed in a UML class diagram (as in Figure 4-21) or directly in a suitable interface definition language.

- A definition of the dynamic behaviour between interfaces and external elements, expressing any temporal dependencies between separate operation invocations. This may be expressed in terms of UML sequence diagrams or collaboration diagrams (Figure 4-28 is an example of the latter).

A mapping should exist between model elements in the Analysis Model and the Design Model. This mapping may be a one to one, or possibly one to many in order to accommodate the more detailed level of modelling required in the Design Model. These relationships may also be many to one where designers have consolidated two or more analysis objects into a design object that performs the analysis object's behaviour. The mappings from the Analysis Model's modelling elements to Design model modelling elements are as follows:

- Actors to external entities in the Design Model.
- Control objects to functional components.
- Boundary objects to interfaces.
- Entity object to interface operation parameters.
- Analysis object interactions to corresponding interface operation types, including factory operations for creating and deleting functional components or their interfaces.
- Analysis collaboration messages to interface operations.
- Analysis collaboration diagrams to design interaction diagrams

Realisation Model:

This defines the physical realisation of the design model in terms that support its integration into other models by the Projection's user, including the constraints of any configuration that can be performed by the user. The Projection definition does

not prescribe the notation for this model, though UML deployment and component diagrams may both be useful here.

Verification Model:

This defines the information and procedures needed by the user of the capabilities of the system defined by the Projection in order to ensure that it is operating consistently with its requirements in the environment in which the user has placed it. The Projection definition does not prescribe the notation for this model.

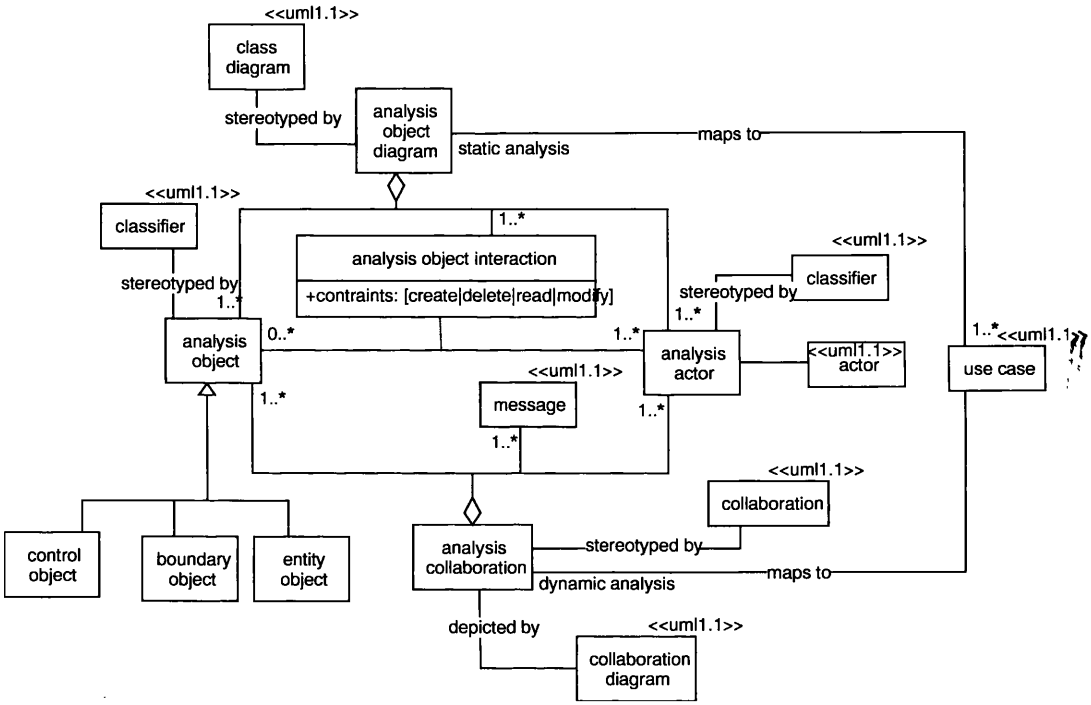


Figure 5-7: Relationship between Elements of the Projection Construct's Use Case and Analysis models

5.2.1.2 Process Guidelines

This section analyses how the notations and meta-model defined for the development framework could be applied usefully to the different development processes defined in section 3.5.

5.2.1.2.1 Generic Development Process

Returning to the Generic SMS Development Stakeholder process model presented in Section 3.5.4 we can see the principle benefits of using the recommendations. By modelling requirements of whatever entity is under developer, i.e. an SMS, a reusable component or an interface standard, in the form of use cases (Recommendation 1) the requirements capture process may be simplified, providing those involved with a common, well-understood expression of requirements. At a minimum the resulting requirements statements would be defined in terms of a use case model, though as discussed in subsequent sections this may be as part of a Business Requirements Model.

By presenting both internal and external existing solutions in terms of Projections (Recommendation 5), the requirements analysis phase becomes a much more homogenised process. The selection of an existing solution may involves comparing the use cases in a Business Requirements Model with the use case models of the Projections' Requirements Model. Another approach to such integration is to refine the Business Requirements Model using UML activity diagrams showing the behaviour of system level or, as shown in Figure 4-34, multi-domain level use cases. As performed in Case Study 5, these activities can then be compared to use cases from a Projection to see where the entity modelled by the Projection (which may be an internally reusable subsystem, a COTS component or an open interface specification) can be applied. This comparison may be supported by mapping functions from the SMS Contracts in a Business System Model to boundary object operations in the Projection's Analysis Model. Alternatively a mapping of Business Process Model Information Flows in and out of the system to entity objects from the Projections Analysis Model may also prove useful in selecting a Projection's subject. Homogenising the form of requirements statements and resulting models therefore eases the selection of solutions within the requirements analysis process for a system. The synthesis of the system's analysis model is also simplified, since much of it may be imported from the analysis models of the existing solutions (Recommendation 4). This feature is already available in CASE tools supporting

UML resulting in the potentially rapid development of an analysis model. Similarly being able to directly import the design models of both internal and external solutions into the overall design model may accelerate the design process. The application of the recommended modelling constructs to the generic SMS development stakeholder process model is summarised in Figure 5-8, indicated by the shaded elements.

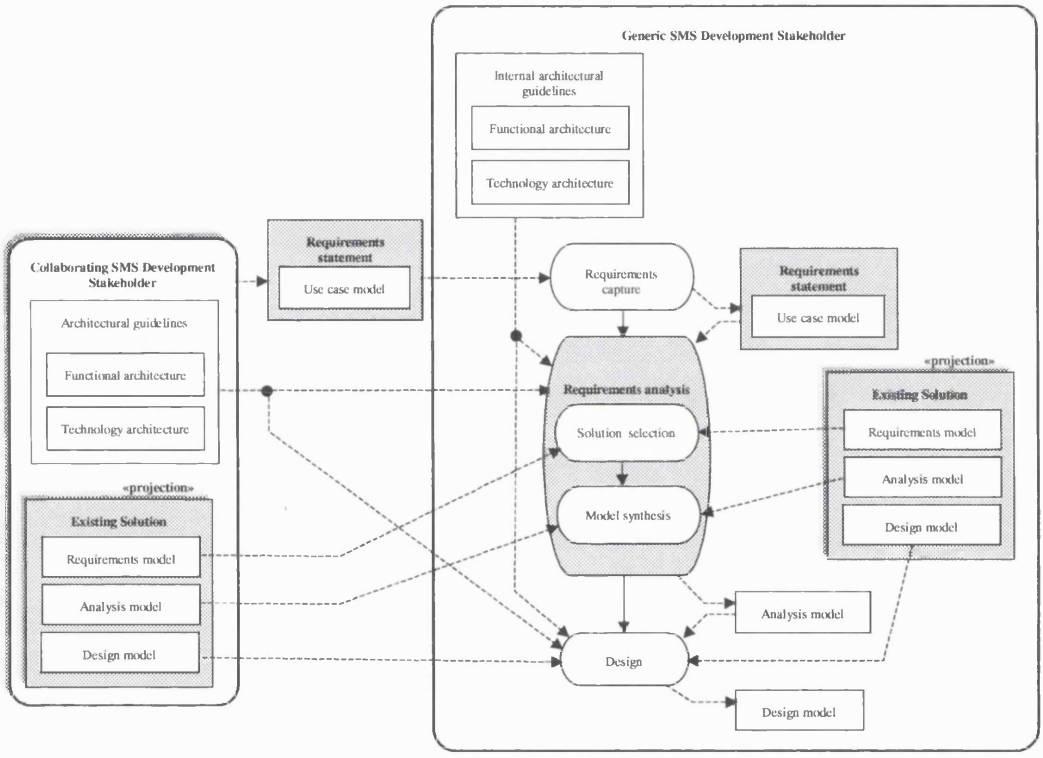


Figure 5-8: The Application of the Methodological Guidelines to the Generic SMS Development Stakeholder Process Model

5.2.1.2.2 Interface Standard Development Process

From the standards development process described in Section 3.5.3, it can be seen how the generic application of the Projection modelling construct is applied in the case of the Standards Developer stakeholder type. As described for the generic case, the requirements analysis and interface design process will benefit from the use of the Projection construct for reuse of internal interface standard definitions and ones

from other standard developers. In support of this, the standard acceptance activity must release new standards in the Projection format for use by others. Though not addressed by these guidelines, standards bodies could gain further benefit from the Projection construct by agreeing the structure of the verification model, so that common approaches could be taken to the ascertaining compliance to an interface standard defined by Projection. It should be noted however, that it is inevitable that standards developers will still have to comprehend the models of existing SMS and NMS systems that are not expressed in terms of Projections.

The Standard Developer stakeholder type is also one of the prime potential beneficiaries of the application of the Business Requirements Model. The other stakeholders are not usually motivated to analyse business situations that involve direct relationships between three or more organisational domains. The SMS Developer for example will not usually need to consider business relationships other than those connecting directly with the Service Provider for which the SMS is being developed. More complex business situations are increasingly appearing however, and it is often left to standards bodies to investigate the ramification of such situations on the standard management functions that need to be supported by individual organisations. One example of this is TINA-C's definition of a multi-player business model. The capability of the Business Requirements Model to represent multi-domain situations and to allow the resulting requirements to be mapped onto requirements at individual Contracts is therefore important. In such cases, Contracts can be used to develop reference points for standardisation. Elements of the Business Requirements Model can be readily mapped to the concepts used in the standardisation mechanisms of TMN, TINA and the TMF as indicated in Table 5-1. This model therefore represents a possible approach to integrating the existing outputs of these bodies and even potentially as a basis for converging their standardisation mechanisms. The impact of the methodological guidelines on the interface standard development model of Section 3.5.3 is shown in Figure 5-9.

Finally, it should be understood that these guidelines do not attempt to provide guidance on how to develop architectural guidelines for standards, but rather assist in how individual functional standards are developed in a way that is more easily communicable with other stakeholder types.

Business Requirement Model	TMN	TINA	TMF
SMS	OSF	Building block	N/A
Contract	Reference point	Reference point	Business Agreement
Use Case	Management service	N/A	Use case
Information Flow	N/A		Process triggers
Business Role	Management service user	Business role	N/A
Responsibility	N/A	Responsibility	N/A
Business Process	N/A	N/A	Business Process
Organisational Domain	TMN	Domain	N/A

Table 5-1: Comparison of Business Requirements Model Concepts and Concepts from the Standards

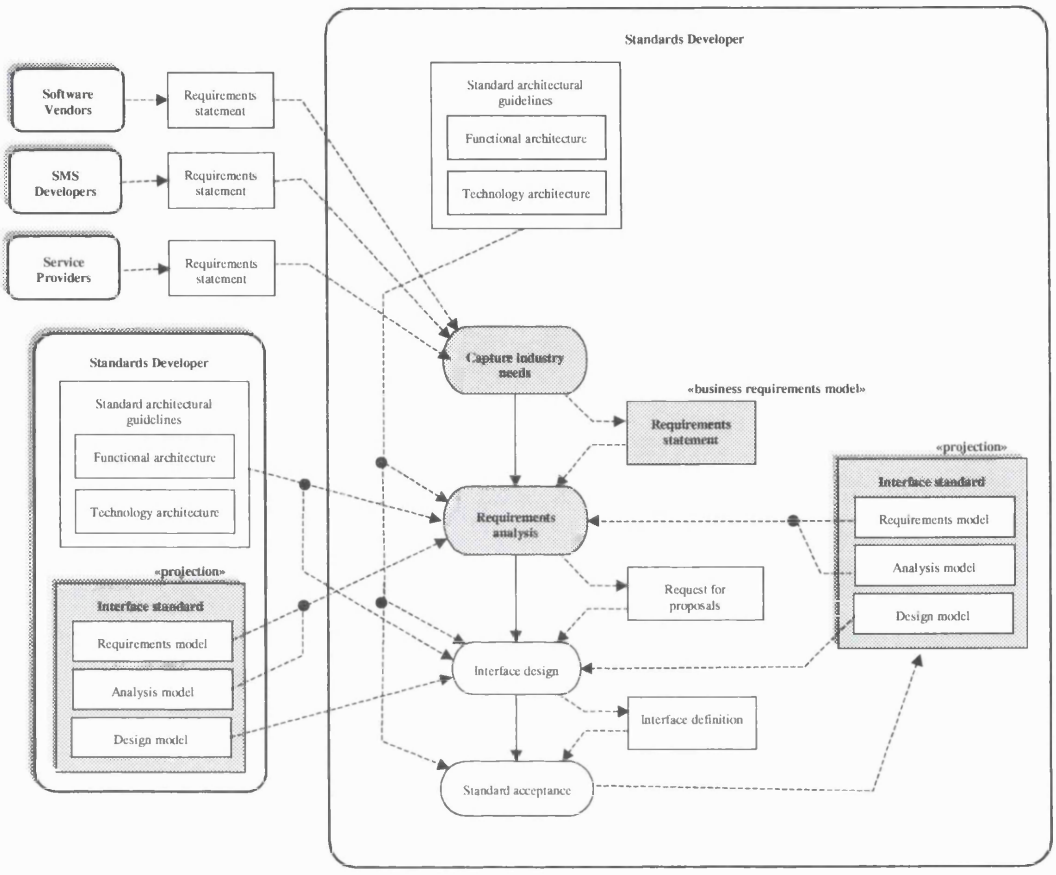


Figure 5-9: Application of the Methodological Guidelines to the Interface Standard Development Process

5.2.1.2.3 COTS Software Product Development Process

Examining the development process for COTS software the Projection construct provides benefits to the Software Vendor stakeholder by providing a common structure both for the interface standards to which products conform and for the existing COTS products maintained by the vendor. For both these sources the Projection's requirements model aids solution selection, its analysis model aids the rapid synthesis of the products analysis model while the design model similarly speeds up the design of the product. The implementation of the COTS product will use items such as libraries from the realisation model of existing products, while the testing of the product will use test cases from the validation model of existing

products and conformance tests from the validation model of interface standards. An important result of using the Projection construct for publishing COTS products is that using other Software Vendor's products in the development of COTS products will be very similar to using internal products. The applicability of the Business Requirements Model to COTS development depends on the granularity of the intended product. However, it is unlikely that the multi-domain aspects of the model will be required as it is expected that a use case model will suffice for documenting a COTS product's requirements.

The product release process can no longer just be concerned with packaging the software for sale and deployment. It must now be concerned with extracting the portions of the models resulting from each of the previous product development processes and selecting suitable subset for inclusion in the product's Projection. Obviously this will be easier if those models use the same notations and meta-models as the Projection, but this is not a mandatory requirement. Figure 5-10 summarises the impact of the application of the guidelines on the COTS development process model.

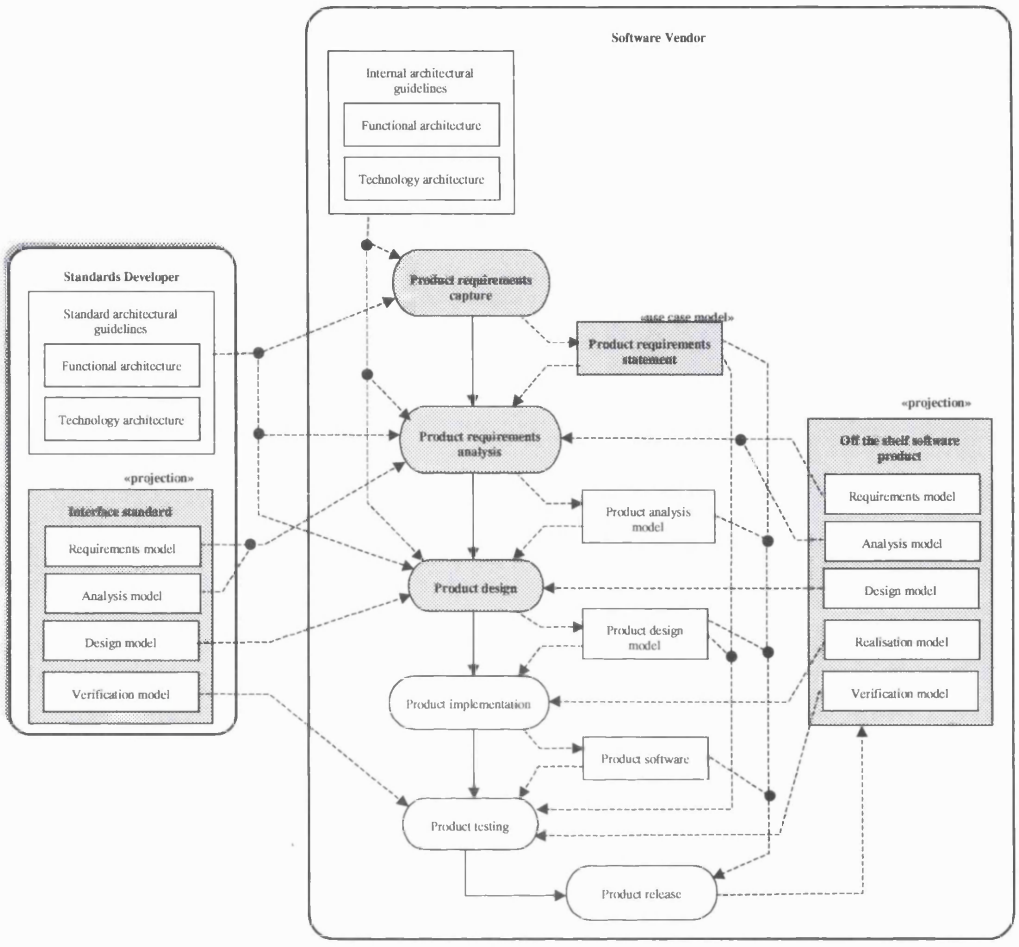


Figure 5-10: Application of the Methodological Guidelines to the COTS Software Product Development Process

5.2.1.2.4 SMS Development Process

The SMS development process is impacted by the use of the Projection construct for interface standards and COTS software products in a similar way to the software development process (see Figure 5-11). The common use of the Projection construct for these external models as well as for the SMS developer stakeholder's internal reusable models offers potential efficiencies in all the development activities. A further activity is introduced, that of SMS release, as it will be necessary to assemble a Projection of the resulting SMS for future reuse. In its most basic form, this documents the whole of the completed SMS within the SMS Developer's domain.

More detailed strategies could be conducted where reusable portions of developed SMS are identified as being particularly suitable for future reuse and documented as separate Projections.

The SMS requirements capture activity needs to capture the business requirements for the SMS and must deal with the inter-domain interactions between the Service Provider and both its Customers and its Third Party Service Providers. It is therefore appropriate to use the business requirements model to express the SMS requirements statement.

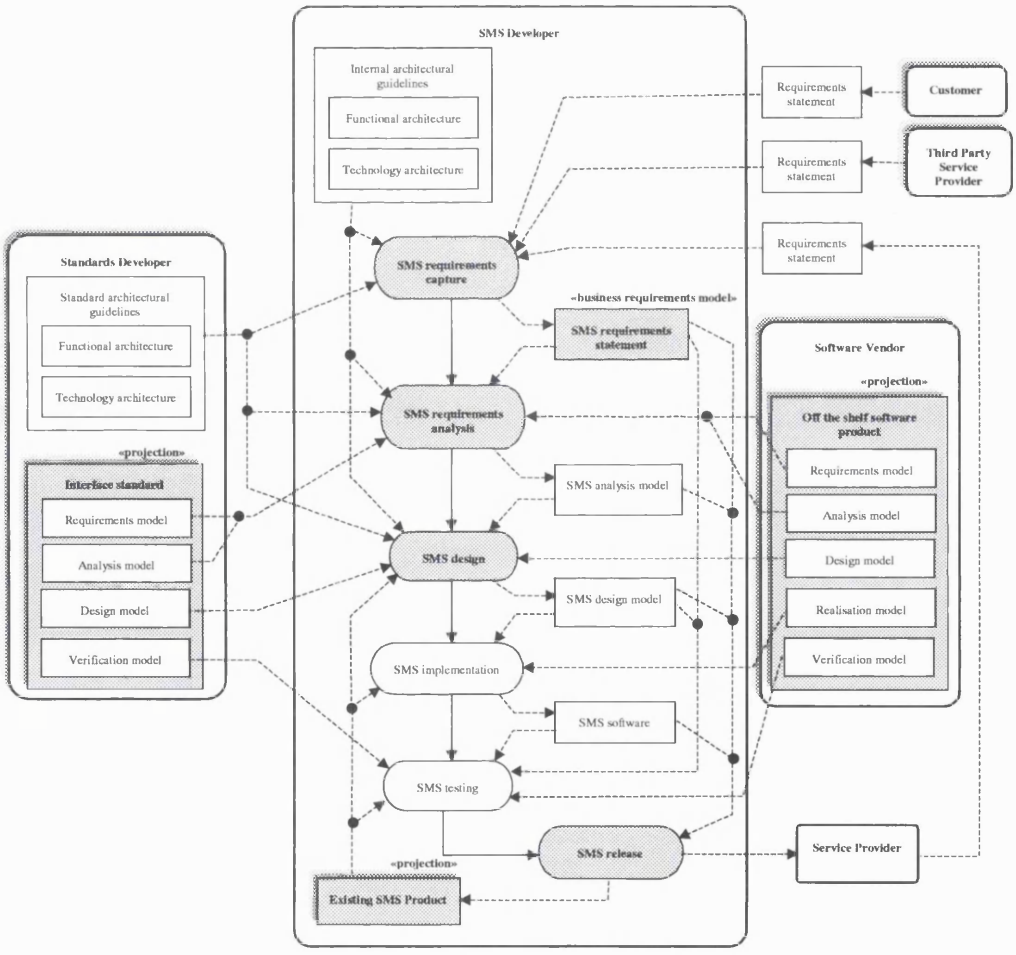


Figure 5-11: Application of the Methodological Guidelines to the SMS Development Process

5.2.2 Architectural Guidelines

The overall approach to defining a development framework has focussed on addressing problems of how SMSs, the standards they conform to and the COTS software they use are developed in relation to each other. The development framework does aim to prescribe a detailed functional framework into which the SMSs, standards and COTS software developed using the guidelines should fit. This has been avoided primarily because the range and volatility of requirements for SMS makes a highly prescriptive functional architecture unlikely to be widely adopted. The final functional structure of the developers' network of SMSs therefore depends largely on the requirements of the Service Providers and the SMS Developers who supply them. The functional structure of the COTS software produced by the Software Vendor will reflect the requirements of specific SMS and what the Software Vendor perceives as being widely marketable functionality. However, in communicating the functional requirements between the stakeholders, some commonly understood architectural framework is useful in order to quickly establish some common terminology and thus reach an agreement on the functional area being discussed. If solutions are structured to fit into such an architectural framework then later requirements can be more quickly matched to existing solutions. Fundamentally, an architectural framework provides its users with a separation of concerns to be addressed in the problem domain, so that specific areas of interest can be consistently identified and other areas not relevant to the problem at hand can be ignored. Such a proprietary architectural framework may prove useful in the context of the products offered by a single SMS developer or Software Vendor. However, a functional architecture will much better match the needs of an open market in SMS software if it is based on some wider industrial consensus and maintained by standardisers in the public domain. Specific solutions that conform to and therefore may populate such an architectural framework should, however, emerge from the industrial stakeholders, rather than being synthesised by professional standardisers. Solutions could be made available either through agreement between several stakeholders, possibly in the context of a standards body (as is the case for the TMF

and the OMG), or published by an industrial stakeholder to present its products and services in a way that conforms to the separation of concerns provided by the architectural framework.

As we have seen in Chapters 2 and 3, there are several open and proprietary software frameworks that are applicable to the problem of SMS development for an open services market. It is difficult to prove whether any of these possess "correct" functional architectures, however the fact that no architecture dominates the market to the exclusion of other suggests that no one of them is superior. To provide architectural guidelines for the development framework presented in this chapter, the main relevant architectures are examined and their concepts judged for compatibility with the methodological guidelines and collected, as appropriate, into a lightweight but consistent architecture.

TMN and its identification of the service management layer underpin the definition of scope of this thesis. The logical layers of TMN are widely accepted as useful separation of concerns by others working in the structuring of open management software, e.g. TINA and the TMF. Combined with the FCAPS categories, TMN provides the functional structure of the 5x5x2 grid shown in Section 2.2.1. However, some of the details of the technology advocated (i.e. the use of OSI management) and reference point categorisation used (i.e. q, x, f etc.) are not so widely used. These are currently under review within study group 4 of the ITU-T [m301x][m301y].

The TMN framework is ultimately aimed at the identification of testable interfaces as functional reference points. As a result it is essentially only a functional separation of concerns and does not directly address the structure of information. Though information modelling is a common activity in TMN system development it relates only to modelling information that will be exposed as an interface to implement a specific management function. TMN does not directly address wider information modelling concerns such as the modelling of corporate data for a whole enterprise.

Another widely accepted model for structuring management information is the TMF's TOM (see Section 2.2.3). This is based on business processes that are

structured in three horizontal layers, the top two corresponding to the TMN SML and the bottom one corresponding to the TMN NML, and three vertical slices into the functional areas of fulfilment, assurance and billing. The TOM contains more information about internal business processes, in terms of information and control flow, than is given by the definition of management functions populating the TMN 5x5x2 grid. The TOM, therefore, gives a better overview of how such management functions relate to the overall structure of the Service Provider's business process requirements and thus provides a better grounding for the generation of requirements at reference points than the TMN grid. As with the TMN, the ultimate aim of the TOM is to locate and identify specific interfaces that are to be standardised, i.e. the architectural framework is provided in order to structure the definition of interfaces rather than the software itself. TINA-C offers a slightly different approach in that it has published detailed component models representing the structure of software grouped according to its own separation of concerns, i.e. service and network architectures and their subdivisions (see Section 2.2.4). However, TINA-C is also ultimately only supporting conformance to interfaces in the form of reference points, rather than the component models behind them.

The TMF provides some further guidance on the structure of management software in its Technology Integration Map. As an analysis of current technologies suitable for management, it suggests the use of: technologies such as Java and WWW browsers for presentation of information; CORBA and possibly workflow technology for business process interaction; CMIPS/P and SNMP for management of network resources and SQL and distributed database technology for access by business processes to operational data. This technology driven approach is very similar to the three tiered architectural approaches currently gaining popularity in the wider distributed processing field. These three tiers are typically referred to as a presentation tier, a business tier and a persistence tier. The presentation tier houses components that deal with human-computer interactions, with web-browser based solutions becoming the norm. The persistence tier houses components that steward specific item of corporate data. The business tier houses components that perform

individual business functions together with business rule driven components (e.g. workflow engines) that co-ordinate the invocation of business functions, user tasks and data manipulation. Three tier architectures are explicitly supported in Java where JavaBeans populate the presentation tier, session Enterprise JavaBeans (EJB) the business tier and entity EJBs the persistence tier [orfali]. The CORBA Component specification [orbos/99-02-05] provides component categories that also map to the different tiers. The three tier separation, as with the TIM, are driven by technological concerns, with the differing needs for platform support for components in different tiers expected to give rise to separate platform product (container) types for components in each tier. In the context of the TIM, interoperability is handled by gateways, which may also have to be integrated into component platforms as discussed in Section 6.3.

The three tiered approach is relatively novel in structuring operational support systems but is important if management systems are to reap the benefits of new component-based distributed processing platforms. Its support for explicit business rules by using workflow techniques in the business tier provides the flexibility required to respond to changing business requirements for service management. Such a three tiered architecture has already been advocated for telecommunications management by BT for its OSS architecture [furley]. Telecordia's long established Information Networking Architecture (INA) and its use in its Operations Systems Computing Architecture (OSCA) also specifies the use of a similar three tiered architecture. Though TINA-C adopted many of the requirements of OSCA/INA and applied them to service management, the three tier, business rule driven approach was not explored, largely due to lack of suitable platform support. Currently the TMF has adopted the OSCA/INA requirements, including the three tier aspects, within its application component team, and is working towards using this as the basis for future standardisation work within the TMF together with the TOM and TMN grid [shrewsbury].

In order to provide the development framework with architectural guidelines that addresses all the stakeholder's needs it is therefore proposed to align the TMN grid

as used by the Standards Developer with its application by the SMS Developer and Software Vendor in the context of a three tiered architecture. It is therefore suggested that within the Standards Developer activities, the TMN grid is used, as at present, to categorise the scope of individual management functions that are defined and whose interfaces are standardised. In other words the TMN grid forms the structure of a repository of standardised management functions that can be used by the SMS Developer and Software Vendor roles. In addition, the management functions should also be mapped to the interfaces between processes in the TMF TOM to provide additional information on the intended applicability of the standardised function, though some management functions will be general enough to be used in many different process interactions. To present their full context however, management functions should be presented using the Projection construct. Typically individual component interfaces or interface segments defined by system's developers may be identified as conforming to a specific standardised management function and will be defined using a profile of that function's Projection. Within the SMS Developer and Software Vendor, the three tier architecture should be used for categorising any reusable components generated. For the SMS developer this should be a natural process if OOSE-based analysis modelling is used in analysing the system's requirements expressed as use cases. The boundary, control and entity class stereotypes map naturally to objects in the presentation, business and persistency tiers respectively. This is one of the major advantages of using the OOSE-based analysis techniques for system analysis in the SMS Developer.

The main outstanding problem with this approach is the standardisation of entities on the persistence tier. Standardised management functions map well to the business process invocations needed in the presentation and persistence tiers, however these functional definitions do not provide good support for the definition of data oriented entities, where data structure, class relationships and relationship constraints are of primary concern. How this may be tackled, especially with respect to the standardisation of data-oriented components is addressed in Section 6.4.

6. Further Work

This chapter discusses further work that may be undertaken or is already planned by the author. This discussion takes into account the work performed for the thesis, the obstacles to practical adoption of the recommendations for a development framework proposed in the previous chapter and new development techniques and system technologies that may have an impact on the further refinement of such a framework.

In particular, the chapter is concerned with:

- The identification of those areas that were analysed but are not fully addressed by the recommendations and how the recommendations may be strengthened through further experimentation.
- How the recommendations may be applied to the emerging standards and architectures for software components.
- How the recommendations may impact on the development of tool support for SMS development and its integration with development techniques for other areas telecommunication software, e.g. IN.
- How the recommendations could be applied to future standardisation in the service management domain.

6.1 Extension and Further Validation of Recommendations

The thesis has been tested through the application of various software development techniques to a number of case studies where representative management systems were collaboratively developed as research prototypes. Evaluation was through a mixture of anecdotal feedback and questionnaires representing the subjective views of development team members as well as through the author's own analysis of the application of the techniques. As pointed out in Section 3.4, this level of rigour is not typical when investigating the effects of software engineering techniques. The large

number of control variables involved in such an experiment, the fact that many are rooted in subjective experience and the high cost of experimental resources (i.e. software engineer hours) makes the cost of experimentation in this area high compared to the confidence that can be expected from results. Nevertheless several further studies could be conducted to improve confidence in the recommendations made by the thesis.

Firstly, the experiment conducted in Case Study 5 featured the development of Projection models for existing components by developers who also were responsible for integrating these components into the required SMS. This serves to confound the developer's assessment of how useful the construct was in documenting the component with their assessment of how useful it was in using the Projection construct in integrating the components into the final system. A more rigorous study would involve SMS developers using component Projections developed by a separate groups of developers. Assessments could then more confidently be made of the following:

- The usefulness of the Projection construct in matching a component's capabilities to the requirements for an SMS, stated in terms of use cases, high-level information requirements and activity diagrams. More specific experiments could be conducted in assessing how useful the construct was in determining, for more complex components, which capabilities and interfaces were relevant and which were not.
- The usefulness of the Projection construct in determining which modifications were required to a component's design based on changes identified to its analysis model. This would require some measurement of how useful the Projection was in communicating the changes in requirements between component reusers and developers.
- The usefulness of the Projection construct in determining secondary modifications needed to a component based on changes required to its design imposed from elsewhere, e.g. changes in the interface design of an interoperating

component. Again this would centre on the use of the Projection for communicating the required changes between component reusers and developers/maintainers.

Ideally such experiments would require the use of CASE tools that directly supported the UML stereotypes and meta-model extensions suggested in the previous chapter.

To increase the confidence in such experiments they would need to be compared to control experiments where similar tasks were attempted between component reusers and developers, but without the benefit of the Projection construct. Ideally, the perception of the developers involved would be complemented with objective measures of the relative effort expended in performing the tasks in each case.

Some extension to the Projection construct can also be envisaged as potentially improving its usefulness to reusers. In Case Study 5, Projection was matched to SMS requirements by comparing aspects of the component's use case based requirements model and its analysis model to the SMS requirements stated in terms of use cases, high level information requirements and activity diagrams. As the activity diagrams typically offer the most detailed view of the SMS requirements, in particular a breakdown of internal functionality, it might be easier to match the requirements to component capabilities if they also were expressed in terms of an activity diagram. Data and events that pass between activities in the SMS requirements could be more clearly matched to data and events exchanged by the component and its environment. This would require the inclusion of a micro process model to the component Projection, which raises questions about how this would integrate with the traceability chain passing through the Projection's use case, analysis and design model. The inclusion of such a micro process model would obviously benefit components destined for the business process tier of a three-tier business architecture. It is less clear how a component destined for the persistence tier might benefit from this. It seems likely that other modelling constructs may be more suitable here, for instance more accurate expressions of the constraints on the

relationships a persistence component may have with other persistence components. How to express this both within the facade Projection and as SMS requirements is an area for further study, with the Object Constraint Language [ad/97-08-08], which is used in defining the semantics of OMG MOF-compliant model, being a possibly suitable approach.

In [jacobsen97], Jacobsen identifies the concept of a facade's variability mechanism. This represents the collection of mechanisms, e.g. inheritance, templates, composition, by which reusers may modify the interfaces and behaviour of a component within limits set by the component developer. Such variability mechanisms were not studied as part of the Projection construct, and their expression using UML needs to be examined further if the construct is to be suitable for practical application in the flexible reuse of components.

One of the main challenges facing the developers of reusable components is the selection of granularity of components. Typically the component must represent a useful level of functionality to the re-user. By packaging the component with its analysis model, this level of functionality is clearly expressed by the set of use cases from which the facade's analysis model is derived. A component should only have loose coupling with other components, with consideration given to merging tightly coupled components into one. Commercial consideration may obviously play a role here, with component vendors being tempted to design components that encourage the user to buy others in a family due to close coupling. Granularity issues have not been addressed in this work as the components used were predefined by others ,e.g. TINA-C, however the analysis model presents a good potential candidate for assessing granularity consideration and applying relevant heuristics. Such heuristics could be driven by integrity concerns and applied using complexity measures of the analysis model.

The definition of the Business Requirements Model defined in Section 5.2.1.1.2 could also benefit from improvement. In particular the relationships between some modelling elements are defined only informally, in terms of how they might be

related though dynamic modelling. This dynamic modelling aspect should be introduced into the meta-model.

6.2 Application to Component Software Architectures

As discussed in Section 5.2.2, component based reuse is seen as an increasingly important approach to accelerating software development, both within the telecommunications industry and in the wider IT community. Building systems from components that interact through well-defined interfaces offers a route both to reusing software across projects of a single SMS Developer and to integrating COTS software from separate Software Vendors. Emerging standards such as Enterprise Java Beans (EJB) [orfali] and CORBA Components [orbos/99-02-05] are promoting the development of platforms that directly support the integration of multi-interface components through the provision of component container platforms. These provide support for remote component interface operation invocation, notification flow between components, directory based naming, persistency, transactions and security. Components also provide support for software deployment, runtime profiling and reflection. However, many existing architectures, such as TMN, do not directly support component-based systems, and not all the notations and tools currently used in telecommunications can fully represent component abstractions for all development activities.

UML currently provides some support for component modelling in the form of component diagrams. However, as identified in Case Study 4, multi-interface components are not directly supported as communicating entities in collaboration or sequence diagrams. There is also no direct support in the UML model of a component for modelling the server interfaces or event sources required of other components. Such additions to UML would assist in making the Projection construct map more directly onto component models such as CORBA Components and EJB. In addition, the currently missing variability mechanism could be structured to be able to map directly to EJB profiles or CORBA Component package profiles. These

issues may well be addressed in responses to an OMG RFP on a UML profile for CORBA [ad/99-03-11], which covers CORBA Components.

TMN functional entities, such as NEFs, MFs, QAFs and OSFs, are not modelled as multi-interface components. However, agent interfaces are often structured in terms of management functions that may be used by separate managers playing different roles. Case Study 2 gives an example of how specific parts of an agent containment tree can be earmarked for use by different managers by placing a manager-role specific MOs high in the tree. By using such role-specific MOs to indicate the demarcation between different access control settings to different agent sub-trees, it may be possible to design MIBs which provide the benefits of multiple interfaces. Alternatively, some of the mechanisms used to manage MO domains for policy based management may be applied [alpers][sloman]. Apart from the lack of support for multiple interfaces, the TMN set of standards already offer many of the same features as components models, including:

- The event report management model [x734], which is close to that used in CORBA Components and EJB event sources,
- Management functions for supporting a management system's repertoire, definition and instance knowledge [x750], which provide reflection support.
- The integration of X.500 and X.700 [bjerring94a], which provides similar location features to component and interface finder interfaces, especially when combined with scoping and filtering.
- Security, which is available in the form of access control MOs [x741] as demonstrated in [gagnon].

Aspects of component models not currently addressed in TMN standards are transactions, persistency and properties. The latter could potentially be handled by custom MO classes, while proprietary solutions for transaction and persistence already exist on commercial management platforms. Given that a TMN component model could be synthesised, a further avenue of investigation would be to develop

specifications for suitable CORBA 3.0-to-TMN and EJB-to-TMN gateways, possibly building on the existing JIDM definitions. The development of a CORBA Component to TMN gateway function has been suggested by the author as a part of a collaborative research proposal.

6.3 Integrated Tool Support

Service providers are addressing competitive pressures through the increased integration of the many software systems that they operate. This includes amongst other, the integration of different OSSs, the integration of OSSs with service access and control software and the integration of service control software for different services that are combined to provide new services.

This thesis argues that a suitable development framework for SMS could be based on common methodological guidelines containing a common meta-model and notation extended from UML. However, when considering the integration of SMS into the wider range of telecommunications software, problems arise from the range of standardised notations used across different specialised areas of telecommunication software development, e.g. SDL in IN-based system development, ODL and IDL in DPE based telecommunication software development and GDMO in TMN-based system. These different languages overlap in some of their features, but also have unique features that maintain their importance in the development and integration of telecommunications systems, e.g. the use of SDL for simulation and test case and code generation. Furthermore, the standardisation of these languages has prompted their support in individual CASE tools. Developers having to develop and integrate systems across the whole range employed by a service provider are faced with a large number of different modelling languages, modelling constructs and modelling tools with only limited integration existing between them [valiant]. This fragmentation in modelling method, notations and tools poses a tangible barrier to the efficient integration of telecommunication systems and the adoption of an open development framework for SMS and other telecommunications areas. Though, as we have seen in Chapter 3, there has been

some work aimed at inter-working between notations and tools (e.g. UML to GDMO, IDL to GDMO), the emergence of similar but separate modelling standards such as the ITU-T's ODL and the OMG's CORBA component IDL extension, indicates that these barriers continue to persist. For this reason, the evaluation and application of interworking approaches between those different notations and their accompanying meta-model needs to be addressed.

A suitable approach should acknowledge that the different branches of telecommunications software development, principally TMN and IN, exhibit differences in development techniques based on real-time requirements, scaleability and reliability targets, installed base, notations used in standards and development tools investment. However, a less parochial approach to the integration of notations and tools would allow future investment in skills and tools to be more widely applicable within enterprises developing telecommunication software. Any fruitful avenue of investigation would have to support the construction of systems from reusable components, even if they are expressed and manipulated by developers in different open notations. Such an approach would also need to support a component's potential involvement in different styles of development, from high level, iconic service creation, to business process engineering led development to more traditional OOAD. Such an approach would aim to facilitate the collaboration of a wide range of telecommunications software development specialists using specialised CASE tools, but interchanging models via component repositories as depicted in Figure 6-1.

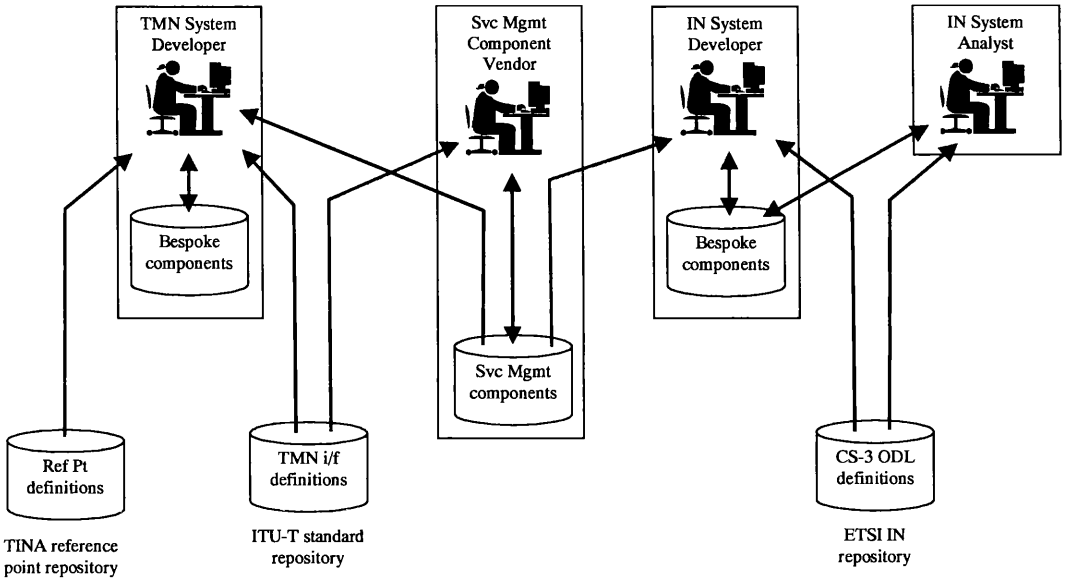


Figure 6-1: A Potential Scenario for the Integration of Software development across the Telecommunications Domain

An approach based on a common, component-based meta-model would be a natural extension of the results of this thesis. Such a meta-model would contain additional information and relationships to enable transparent roundtrip transformation to the notation most suitable for different telecommunications development tasks, e.g. UML for general analysis and design work, SDL for behaviour simulation or test case generation, IDL or GDMO for deployment on specific distributed platforms.

It is not envisaged that a single homogenous, single-vendor tool set will ever address the entire telecommunications development domain. Instead research should aim to ensure open model interchange between both general purpose tools, such as component repositories and UML editors (though working with telecommunication specific profiles), and specialised telecom-specific tools such as GDMO compilers or SDL-based service creation simulators. The OMG's Meta Object Facility (MOF) [ad/97-08-14] and XML-based Model Interchange (XMI) standards [xml][ad/98-10-05] would be central to such research. The use of XMI would allow tools and exchange mechanisms to gain leverage from the large range of software emerging that support XML ,e.g. MicroSoft's Windows 2000, thus potentially reducing the

cost of supporting an open model exchange format. It may also allow XML to form the basis of open consistency management tools needed to ensure cohesion of models being manipulated in different native notations and mapped to and from the component meta-model, as depicted in Figure 6-2.

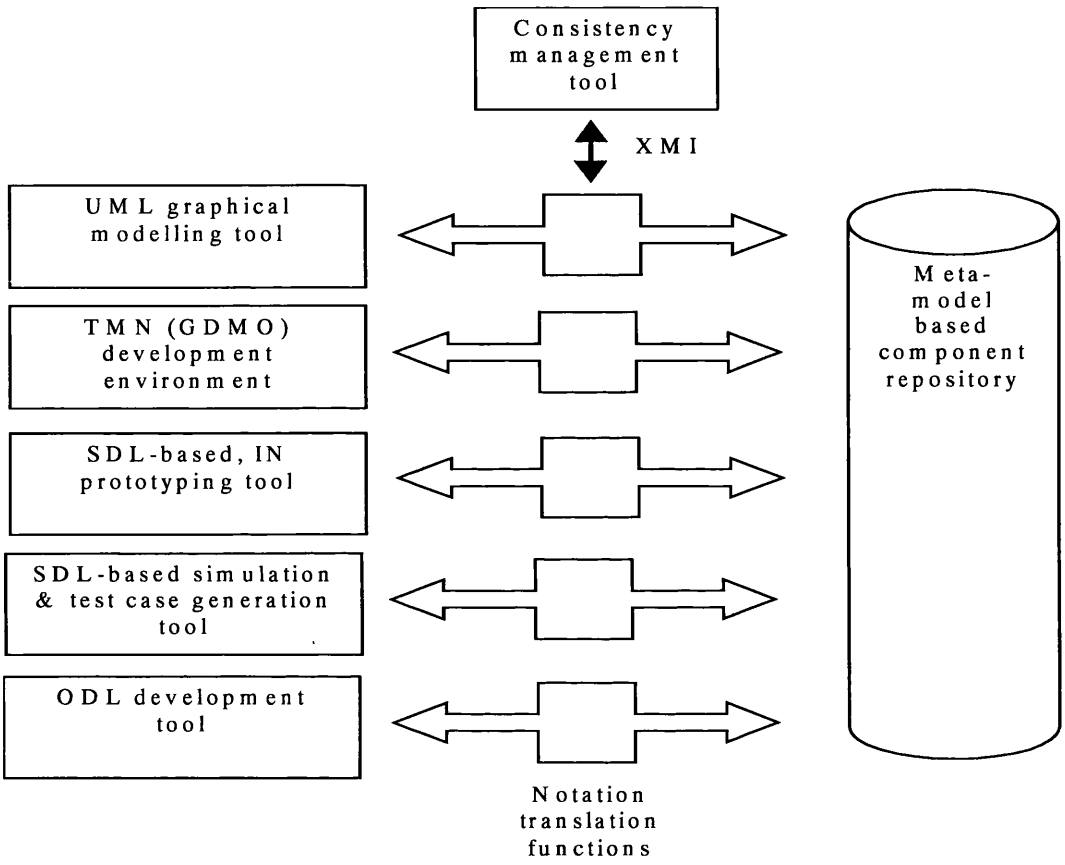


Figure 6-2: Tool interworking and multi-notation round trip engineering using XMI

Such as approach, based on the author's ideas of a common, multi-notational meta-model and model exchange mechanisms, have formed the core of a collaborative research proposal recently submitted to the European Commission.

6.4 Application to Service Management Standardisation

The approach taken in this thesis to addressing the development of open management systems has focussed on addressing the process and concomitant notations of open system development. It has not promoted the detailed definition of

a functional architecture for service management or the specification of functions that might populate such an architecture. This is based on the assertion that the business requirements on service management are currently too volatile. However, the author acknowledges that a methodologically-based approach will not, in itself support open service management, and that the requirements of service management will have to be addressed by standards bodies working in this area in spite of the changing requirements present. This section presents some changes in service provision that, due to their general industry momentum, may be fruitful targets for service management standardisation. It also addresses how various changes in technology impact on the common perception of the scope of service management and speculates what further development may have profound impacts on the development of service management functions.

One major area impacting on telecommunications is the rapid expansion of the Internet, and the resulting move by service providers to offer IP services instead of or alongside traditional connection oriented services such as PSTN and ATM. The use of IP technology presents several challenges to service providers used to dealing with traditional telecommunications services such as IN or ATM. In these cases there is a clear identification of the point at which a service is provided to the customer, e.g. a User-Network Interface (UNI). This, therefore, provides a point at which service performance can be monitored for application to terms in an SLA. For IP-based service the situation is different. The IP-based services that users are familiar with reflect the behaviour both of the network and of end system processes such as user applications and application servers, e.g. for email store and forward, WWW or security keys. This reflects the engineering tendency in the Internet to push intelligence to computer systems at the edge of the network, compared to concentrating it within the network as in telecommunications systems.

As we have see in Chapter 3, the telecommunications industry has a well-established functional architecture for structuring management in the form of TMN. The management of IP-based networks does not benefit from any such common functional structuring, and little work has been done in formalising the application of

the TMN functional grid to such networks despite the fact that they are within the scope of TMN. With the introduction of QoS capable IP networks (based on RSVP and Diffserv) there is, as observed in Chapter 3, increased interest within the IETF in address management beyond the network element. Additionally the Distributed Management Taskforce is working towards a Common Information Model (CIM) for managing the end systems typically deployed in IP-based enterprise networks. Neither group has aligned its work with the TMN functional architecture. This is necessary if telecommunication service providers are to apply themselves in an integrated way to the provision of IP enterprise network management services.

Strongly influenced by the rise of the IP technologies, enterprises are increasingly using distributed IT systems based on Intranet technologies to support and integrate their business processes. Two major trends are visible in the development of IT support for integrating enterprise business processes, the move to component-base, three tier architectures and the adoption of workflow management techniques. However, if service providers wish to exploit these trends, they must be able to offer services that manage of the networks that underlie such business integration and the customer's business processes that interact across them, i.e. they should provide third party business process management services. Such services would also aim to support business process automation between enterprises as well as within them, based on multi-domain Intranets, also known as Extranets.

The requirements to manage IP-based services and, more specifically to manage three-tier architecture-based enterprise management systems, means that the management of services will have to be integrated with what is more commonly known as systems management. The systems being managed would include distributed application servers (e.g. middleware service servers and workflow engines) as well as the management systems themselves. Current management architectures, such as TINA and the TMF's Telecoms Operation Map, include systems management but typically only as a single, orthogonal functional area, typically lacking the finer grained guidance on separation of concerns that is given

for communications systems. Solutions also often focus on computing hardware management rather than the management of the software processes.

Figure 6-3 represents a possible system architecture for an IP-based enterprise management system. The principal architectural concept used is that all parts of the system are modelled and implemented as software components, which have an underlying technological grouping into either a presentation, business or persistence tier. Starting from this basis an open functional architecture is therefore defined in terms of groups of component types addressing different functional concerns, along lines similar to those outlined in Section 5.2.2.

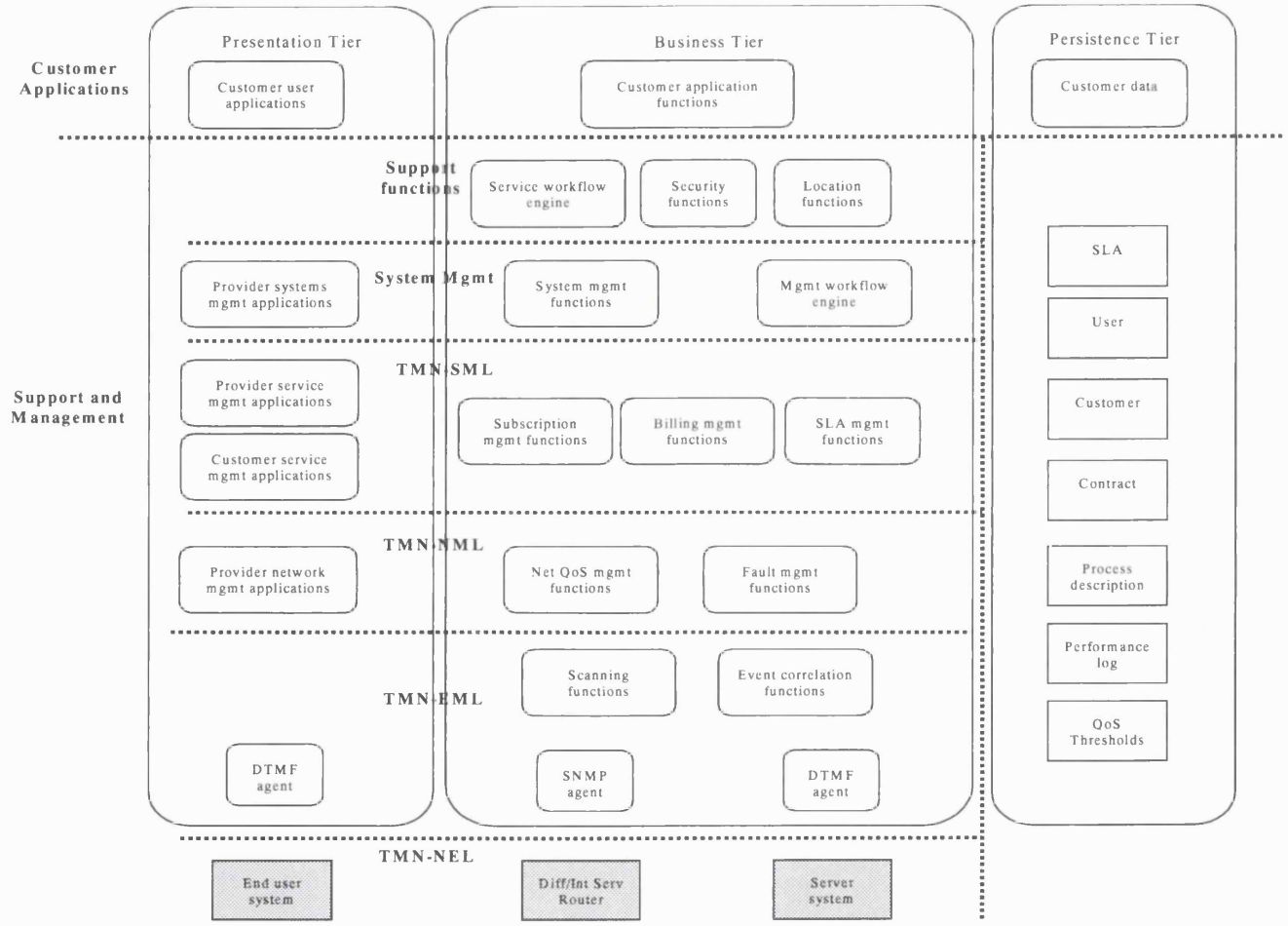


Figure 6-3: Possible enterprise management service system scenario showing potential functional architecture overlay

Figure 6-3 raises many of the architectural issues presented by the move to a three tier software architecture for integrating network, service and systems management with a TMN framework. The key questions raised are:

- What is a suitable functional decomposition for components in the persistence tier?
- Should systems management components be treated separately from service management ones?
- Should non-network hardware systems be managed in the same functional area as network elements?
- How can existing network element MIB definitions (e.g. in SNMP) and useful NE management functions (e.g. workload monitoring) be integrated into a component-based software architecture?
- Should the component groupings used for forming and populating such a functional architecture be the only groupings considered, or should other, potentially overlapping groupings based on component deployment, lifecycle or security management, component fault and performance monitoring or product marketability also be supported in an open fashion?

The mechanism for the management and monitoring of application components is another related area in need of further investigation. In addition to the work of the DTMF CIM, requirements for this have already been addressed by the TMF Application Component Team, which has produced requirements for a Common Application Management Interface as part of its generic building block requirements [shrewsbury]. This in turn has influenced an OMG RFP on the subject [orbos/99-04-11].

The approach required to perform the on-going standardisation of service management functions in such functional architecture also requires some research. One possible approach is that taken by the BT OSS architecture, which assumes that corporate data is more stable than the business processes that operate on it.

Standardisation efforts could therefore focus on defining common groupings for components in the persistence tier. Such common persistence objects would capture the structure of units of common corporate data, the operations that could be performed on them and the constraints on relationships to other objects in the persistence tier. It is significant that the TMF in its current TOM work ignores the definition of data and focuses primarily on business processes, which provide direct requirements for business tier objects only. At the time of writing the author is assisting the TMF's Application Component Team, in examining the differences in the TMF's process-oriented approach and the BT data-oriented approach. This work has the aim of influencing the TMF to change its working structure to one that combines both approaches and that addresses each of the three tiers explicitly.

Whichever approach is taken to developing a suitable functional architecture for service management, the requirements which act upon it will have to be regularly reviewed as the common understanding of the meaning of service management changes. The EURESCOM project P.812 is a good example of such a review [davidson99a], however it may be necessary for standards bodies such as ITU-T SG10 to regularly undertake analysis of evolution scenarios in order to be able plot the direction of the scope and content of the service management layer. Examples of evolution scenarios that are relevant at the time of writing could address:

- The method by which a user pays for services moves from subscription based quarterly payment based on standing charge and usage metering, through "pay as you go" schemes, to anonymous, per session e-cash/micropayment transactions.
- The evolution of voice services from PTSN, through PSTN-IP integration to a scenario of pure voice over IP as part of IP integrated services. This would involve the migration of much of the intelligence that supports intelligent network from the network to end user devices.
- A move to the situation where the relative costs of bandwidth and network intelligence are such that only premium services are cost effective to charge for

and a large number of services are provided at a flat rate or free to users (e.g. the current large number of free ISPs in the UK).

- The migration to a situation where communications services are highly commoditised, charging policies are highly dynamic and users may change providers on an hourly basis, possibly with the help of intelligent brokering agents.
- The migration to a situation where users expect to be able to request whatever SLA they require at the time, possibly on a call by call basis.
- User's subscribe and unsubscribe rapidly while roaming internationally through domains of local area, high bandwidth access providers such as train, aeroplane, airport, hotel and highway operators, with closely linked loyalty schemes for these non-communication service providers.

Such analyses could be conducted so as to order requirements in terms of their volatility, with the most invariant requirements being those most directly targeted for ongoing standardisation work, e.g. through the definition of service management persistent object. This approach has been suggested by the author in a further collaborative research proposal that has recently submitted to the European Commission.

7. Conclusions

This chapter reviews the conclusion of the thesis and discusses the extent to which the recommendations for the proposed open development framework satisfy the goals laid out in Chapter 2 and the thesis statement in general.

The analysis of the current state of Service Management standards in Chapter 2 revealed that no single logical architecture is dominant. The current candidates differ widely in their approach, from the coarse grained, relatively unpopulated TMN service management layer, through the business process driven set of TMF standards to the TINA reference points generated from a DPE-based component model. In addition, a wide range of technologies was identified in Section 2.3 as being applicable to service management. This included CMIS, as currently specified in TMN (though this is not widely regarded as a suitable technology for service management) and CORBA, due to its general suitability for distributed business systems. This heterogeneity in both logical and technological architectures for service management presents developers with increasing problems of interoperability between SMSs precisely at a time when open markets in telecommunication services are increasing the need for integration of service management processes between service providers and their customers. The thesis therefore suggests that, to improve the chances of building interoperable SMS within increasingly tight cost and time constraints, a suitable development framework exhibits the following characteristics:

- It adopts a loose logical architecture in order to accommodate the range of logical structures currently applicable to SMS.
- It exploits the existing and emerging range of technological gateways to accommodate a wide range of SMS platform technologies.
- It promotes interoperability and efficient SMS development through a common set of methodological guidelines.

The thesis has shown that SMS development can benefit from an open approach to modelling and integrating SMS systems that is commonly understood by SMS Developers, COTS Software Vendors and service management Standards Developers. An analysis of the different software engineering and modelling techniques that may be applied to the development of telecommunications management systems revealed the potential linkages between the development activities within each of the relevant stakeholders. These linkages are identified as points of communication where a common development framework would be most beneficial.

The thesis asserts that existing network management development techniques are insufficient to form the basis of such methodological guidelines. Chapter 3 reveals the distinction, observed by many standardisers and developers, between the modelling of interfaces and the modelling of systems, the latter of which consists both of interface definitions and models of the systems behind them. The needs of the SMS Developer stakeholders depend on the *smooth integration of interface and system modelling* techniques. This was hindered in the case of M.3020 due to its reliance on functional decomposition, which made it difficult to exploit the benefits of OOAD in terms of software and specification reuse and specialisation. Though M.3020 is currently being revised to use UML [m3020-99], it still adheres to this functional decomposition approach, thus maintaining its inadequacy for SMS development.

A series of case studies was conducted around the development of SMS in a number of research projects. These provided evidence of the development techniques that developers found most useful in practice. Based on this evidence and the analysis of current development techniques used in management system development, the following general recommendations were made:

- *Use case modelling* should be used for describing the external functionality of service management standards, systems and components.

- For multi-domain SMS analysis, *business roles* and *business processes* should be used to supplement use cases.
- The *UML notation* should be used both internally for the different stakeholder development processes and externally for exchanging models between developers involved in these processes.
- The *Projection modelling construct* should be used for publishing COTS software, for publishing standards and for documenting internally developed reusable software.
- Where possible, an analysis and design process that uses *OOSE analysis modelling* should be adopted.

These recommendations are made concrete in the form of semi-formal descriptions of UML modelling constructs, namely the *Business Requirements Model* and the *Projection Model.* These represent the main contribution of the thesis. Some examples of their application are given with respect to the models used in Case Study 5.

To assess the correctness of the thesis statement, the methodological guidelines and supporting evidence from the case studies and state of the art analysis are assessed below against the development framework goals defined in Section 2.4.

> ***Goal 1:*** *The Development Framework must support SMS Developers in developing a SMS that satisfy the business needs of Service Providers, including its business interactions with Service Customers and Third Party Service Providers.*

The guidelines provide a Business Requirements Model which it recommends is used in the requirements capture activity of the SMS development process. This model supports techniques of business role and responsibility modelling to determine the requirements for interactions between the service provider's SMS and the systems of its customers and collaborating service providers. These have been found useful when combined with sequence diagrams in the requirements analysis

stage (Case Study 2, 3, 4 and 5). In addition this model supports the identification of business processes that may occur internally to the Service Provider. This was found useful in Case Study 5.

> ***Goal 2:*** *The Development Framework must address all stages of SMS development, i.e. requirements capture, system analysis, system design, systems testing, system deployment and system maintenance.*

Though the process models for the SMS Developer stakeholders identify activities for all stages of development, it has only been found to be practical to define common methodological guidelines for the requirements capture, the requirements analysis and, to an extent, the design activities. Some aspects of the design model as well as the models for implementation, testing and deployment are very dependent on the technological and programming environment adopted, so defining common guidelines becomes problematic. System maintenance is treated simply as a further iteration of the overall development process, though it is acknowledged that this may not be the most efficient approach. System maintenance has not been addressed in the case studies.

> ***Goal 3:*** *The framework must support SMS Developers in the application of open standards from Standards Developers.*

This situation was tested in Case Studies 3, 4 and 5 where open specifications, primarily ones originating from TINA, were followed in order to implement various systems. This experience revealed some shortcomings in the use of ODP viewpoints (Case Study 3) but also showed how re-documenting the specifications, first with UML and then using an equivalent of the Projection construct, aided in their smooth application by SMS developers. Case Studies 1 and 2 also supported the evidence from the state of the art review that indicated that specifying open interfaces using functional decomposition did not support the needs for such definitions in the SMS development environment. Instead, a use case or scenario focussed approach is advocated. In addition, Case Study 5 supported the recommendation that the analysis

level specification for an open interface should be structured using OOSE analysis object types.

> ***Goal 4:*** *The Development Framework must support SMS Developers in the reuse of design solutions and software over different projects.*

The recommendation to use the Projection construct will aid in the reuse of internally developed models, especially with respect to the retention of the original requirements and their traceability to design model elements. This was demonstrated by components such as subscription management, which was reused and modified over several of the case studies with the aids of some of the modelling elements that make up the Projection.

> ***Goal 5:*** *The framework must support SMS Developers in using commercial off the shelf software developed by Software Vendors.*

Though the case studies involved some existing components, these were subject to extensive remodelling using the recommended techniques, so were not really "off-the-shelf". This situation was simulated however in Case Studies 3 and 4 where components from one organisation were reused unchanged by another, thus providing some evidence of the validity of the recommendations in addressing the goal of reuse of software between organisations.

> ***Goal 6:*** *The Development Framework must support the Software Vendors in the application of open standards in developing its products.*

This was supported in much the same way as for Goal 4. It should be noted that some of the components reused by other partners in Case Studies 3 and 4 were based on open interface specification as part of component specification of the TINA Service Architecture, e.g. the subscription management component.

> ***Goal 7:*** *The Development Framework must support Standards Developers in the on-going development and evolution of open standards to be used by SMS Developers and Software Vendors.*

The fact that existing standard specifications, such as those from TINA, could be remodelled and applied successfully using the recommended techniques indicates how existing standards can be aligned with these techniques, and thus, not insignificantly, with each other. The full benefit of these techniques clearly only will come when the relevant bodies adopt similar techniques for their internal development work. In the case of the recommendation to use UML, the TMF has already adopted it while TINA-C has its adoption as a stated aim. A recent draft revision to M.3020 [m3020-99], shows that the ITU-T is also moving to use UML more widely. This revision specifies the use of use cases for defining management services and functions and explicitly applies the requirements capture, analysis and design stages of systems modelling. However, differences will remain in the way the different bodies structure their standardisation efforts, with the ITU-T defining management functions derived from management services, the TMF defining management functions based on business process interactions and TINA-C defining reference points based on component models. The recommended Business Requirements Model accommodates all of these analysis approaches, thus providing a possible route to converging the different standardisation approaches. Also, the Projection modelling construct provides a common form for expressing the results of standardisation so that the differences in the approach used in arriving at a standard may not present such an obstacle for the standard's user.

> ***Goal 8:*** *The framework must support the development of a SMS that operate over heterogeneous computing platform technology and which will be robust to changes in computing platforms.*

This goal is addressed by adopting a common notation and meta-model for requirements capture, analysis and design within the Projection construct, regardless of the target technology. At the requirements capture and analysis level, it is expected that models will remain unchanged during platform technology changes. It is expected that aspects of the design model may change if the platform changes, especially when changing paradigms, for instance when moving between client-server and manager-agent platforms. Such 'protocol neutral modelling' is not, in the

author's opinion, possible at this detailed level of abstraction. However, the same UML-based modelling techniques can be applied, with platform specific notations, e.g. GDMO and IDL, being generated automatically, and thus made largely transparent to the designer. The development in Case Studies 3, 4 and 5 involved both CORBA and CMIP technologies. However, the latter did not form a large part of the development effort, and was restricted to the network and element management layers. So, though the views of the CMIP developers when questioned did not vary much from those of the CORBA developers, support for claims that the techniques recommended applied equally to both is relatively weak. One group of developers who modelled their system in Case Study 2 using OMT, were however able to transform this CMIP-based design to a CORBA-based one for Case Studies 3 and 4 using many of the same core concepts.

> ***Goal 9:*** *The notations and methodology of the development framework should be easy for those playing SMS development stakeholder roles to understand, and should be readily supported by CASE tools.*

By adopting the widely used UML notation and the use case modelling construct, and by avoiding the use of FDTs and of the more complex aspects of ODP enterprise modelling, the recommendations aim to be accessible by the widest range of developers in the SMS development arena. The use of UML ensures widespread CASE tool support for the implementation of the recommendations. Expressing the recommendations in terms of a meta-model defining a UML profile may promote the development of CASE support for SMS in existing tools.

To summarise, therefore, this thesis has provided evidence of the need for a loose logical and technological for SMS development and thus of the potential benefits of adopting a common development framework that can cope with such heterogeneity. A set detailed goals for such a framework and a model of the development process needs of the relevant stakeholder types were established. A series of case studies was conducted which garnered both informal and empirical evidence on the usefulness of a range of software development techniques applied to SMS development. Based on

the framework goals and stakeholder process needs, specific methodological recommendation have been made and supporting notational models have been defined, principally the Projection Construct and the Business Requirements Model. The author expects that these recommendation and notational models may have applicability beyond SMS development, but providing evidence for this is beyond the scope of this work.

This page has been deliberately left blank.

8. References

[ad/97-08-03] UML Summary, v1.1, ad/97-08-03, OMG, Aug 1997

[ad/97-08-04] UML Semantics, v1.1, ad/97-08-04, OMG, Aug 1997

[ad/97-08-05] UML Notational Guide, v1.1, ad/97-08-05, OMG, Aug 1997

[ad/97-08-06] UML Extension for Objectory Process for Software Engineering, version 1.1, ad/97-08-06, OMG, Sep 1997

[ad/97-08-08] Object Constraint Language Specification, version 1.1, ad/97-08-08, OMG, Sep 1997

[ad/97-08-14] Meta Objet Facility, revised submission, ad/97-08-14, OMG, Aug 1997

[ad/98-10-05] XML Metadata Interchange, ad/98-10-05, OMG, Oct 1998

[ad/99-03-11] RFP: UML Profile for CORBA, ad/99-03-11, OMG, 1999

[adams] The Lean Communication Provider: Surviving the Shakeout through Service Management Excellence, Adams, E., Willetts, K., 0-07-070306-X, McGraw-Hill, 1996

[allen] Putting UML to Work: Strategies and Techniques, Allen, P., in [UML98], pp33-43, OMG, Jun 1998

[allweyer] Process Orientation in UML through Interaction of Event-Driven Process Chains, Allweyer, T., Loos, P., in [UML98], pp183-193, OMG, Jun 1998

[alpers] Concepts and Application of Policy Based Management, Alpers, B., Plansky, H., in [IM95], pp57-68, Chapman-Hall, 1995

[arkko] Requirements for Internet-Scale Accounting Management, Arkko, J., draft-arkko-acctreq-oo.txt, IETF, Aug 1998

[arlow] Literate Modelling - Capturing Business Knowledge with the UML, Arlow, J., Emmerich, W., Quinn, J., in [UML98], pp165-171, OMG, Jun 1998

[aurrecoechea] Towards building manageable multimedia network services, Aurrecoechea, Lazar, A.A., Stadler, R., Proceedings of the International Conference on Management of Multimedia Networks and Services, Montreal, Canada, pp18-30, Chapman-Hall, 1997

[barbeau] An Approach to Conformance Testing of MIB Implementations, Barbeau, M., Sarikaya, B., in [IM95], pp654-666, Chapman-Hall, 1995

[barillaud] Network Management using Internet Technologies, Barillaud, F., Deri, L., Feridun, M., in [IM97], pp61-70, Chapman-Hall, May 1997

[basili] The TAME Project: towards improvement-oriented software environments, Basili, V.R., Rombach, H.D., IEEE Transactions on Software Engineering, 14(6), pp758-773, 1988

[beaumont] Starting from Scratch, Beaumont, J, Telecommunications, 26-29 March 1995, Conference Publication No. 404, IEE, pp1-3, IEE, 1995

[berndt95a] Service Architecture, Berndt, H., Minerva, R., TINA Baseline Document TB_MDC.012_2.0_94, TINA, 1995

[berquist] Managing Information Highways: The PRISM Book: Principles, Methods, and Case Studies for Designing Telecommunications Management Systems, Berquist, K., Berquist, A. (Eds), Lecture Notes in Computer Science 1164, Springer-Verlag, 1996

[bjerring94a] Requirements of Inter-Domain Management and their Implications for TMN Architecture and Implementation, Bjerring, L.H., Tschichholz, M., in [ISN94], pp193-206, Springer-Verlag, 1994

[bjerring94b] End-to-end Service Management with Multiple Providers, Bjerring, L.H., Schneider, J.M., in [ISN94], pp306-316, Springer-Verlag, 1994

[bleakley] TMN Specifications to Support Inter-Domain Exchange of Accounting, Billing and Charging Information, Bleakley, C., Donnelly, W., Lindgen, A., Vuorela, H., in [ISN97], pp275-282, Springer-Verlag, May 1997

[booch94] Object Oriented Analysis and Design with Applications (2nd edition), Booch, G., Rumbaugh, J., Jacobsen, I., 0-8053-5340-2, Benjamin Cummings, 1994

[booch99] The Unified Modelling Language User Guide, Booch, G., Rumbaugh, J., Jacobsen, I., 0-201-57168-4, Addison-Wesley, 1999

[bosco] ACE: An Environment for Specifying, Developing and Generating TINA Services, Bosco, P.G., Lo Giudice, D., Martini, G., Moiso, C., in [IM97], pp515-527, Chapman-Hall, May 1997

[caluwe] The Use of TMN as an Architectural Framework for Value Added Service Management, de Caluwe, I., Leever, P., Wester, J., in [ISN94], pp295-304, Springer-Verlag, 1994

[carls] Introducing SDL'92 in the Development of TMN Applications, Carls, G., Frohnhoff, B., in [ISN98], pp365-378, Springer-Verlag, May 1998

[case] Simple Network Management Protocol (SNMP), Case, J.D., Fedor, M., Schoffstall, M.L. Davin, C., RFC 1157, IAB, 1990

[chan-m] Customer Management and Control of Broadband VPN Services, Chan, M.C., Lazar, A.A., Stadler, R., in [IM97], pp301-314, Chapman-Hall, May 1997

[chapman] Overall Concepts and Principles of TINA, Chapman, M., Montesi, S., TINA Baselines document TB_MDC.018_1.0_94, TINA, 1994

[chatt] TMN/C++: An Object-Oriented API for GDMO, CMIS and ASN.1 , Chatt, T.R., Curry, M., Holberg. U., Seppa, J., in [IM97], pp177-191, Chapman-Hall, May 1997

[chen96] Distributed Network Management Using CORBA/TMN, Chn, G., Neville, M., Kong, Q, Proceedings og the 7th IFIP/IEEE International Workshop on Distributed Systems Operations and Management, L'Aquila, Italy, Oct 1996

[chen97a] Integrated TMN Service Provisioning and Management Environment, Chen, G., Kong, Q., in [IM97], pp99-112, Chapman-Hall, May 1997

[chen97b] The Business Process and Object Modelling for Service Ordering, Chen, G., Kong, Q., Proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems Operations and Management, L'Aquila, Italy, pp196-209, 1997

[choi] A Generic Service Order Handling Interface for the Cooperative Service Providers in the Deregulated and Competitive Telecommunications Environment, Choi, Y.B., Tnag, A., in [ISN97], pp211-218, Springer-Verlag, May 1997

[christensen] Information Modelling Concepts, Christensen, H., Colban, E., TINA Baseline document TB_EAC.001_1.2.94, TINA, 1994

[coplien] Pattern Languages of Program Design, Coplien, J., Schmidt, D. (eds), 0-201-60734-4, Addisson-Wesley, 1995

[corba] The Common Object Request Broker Architecture and Specification, OMG Document number 92.12.1, Rev. 1.1, OMG, 1992

[corley] The Application of Intelligent and Mobile Agents to Network and Service Management, Corley, S., Tesslaar, M., Cooley, J., Meinkohn, J., Malabocchia, F., Garijo, F., in [ISN98], pp127-138, Springer-Verlag, May 1998

[covaci] Towards Harmonised Pan-European TMN Customer Care Solutions: Interoperable Trouble Ticketting Management Service, Covaci, S., Dragan, D., in [ISN97], pp255-262, Springer-Verlag, May 1997

[dahle] Method and Graphical Syntax for Computational Modelling, Dahle, E., Giganti, P.L., Proceeding of TINA'95, Melbourne, Australia, pp577-590, TINA, Feb 1995

[dassow] SNMP and TMN: Aspects of Migration and Integration, Dassow, H., Lehr, G., in [ISN97], pp339-348, Springer-Verlag, May 1997

[davidson94] Service Provisioning in a Mulit-Provider Environment, Davidson, R., O'Brien, P., in [ISN94], pp259-271, Springer-Verlag, 1994

[davidson99a] A Practical Perspective on TMN Evolution, Davidson, R., Turner, T., in [ISN99], pp3-12, Springer-Verlag, Apr 1999

[davidson99b] A New Architecture for Open and Distributed Network Management, Davison, R., Hardwicke, J., in [ISN99], pp23-38, Springer-Verlag, Apr 1999

[dede] OSAM Component Model, A key concept for the efficient design of future telecommunication systems, Dede, A., Arsenis, S., Tosti, A., Lucidi, F., Westerga, R., in [ISN97], pp127-136, Springer-Verlag, May 1997

[delafuente94] Management Architecture, v2.0, de la Fuente, L.A. (ed), TINA Baseline Document, TB_GN.010_2.0_94, TINA-C, Dec 1994

[delafuente95] Application of the TINA-C Management Architecture, de la Fuente, L.A., Kawanishi, M., Wakano, M., Walles, T., Aurrecoechea, C., in [IM95], pp424-436, Chapman-Hall, 1995

[deri] Static vs. Dynamic CMIP/SNMP Network Management Using CORBA, Deri, L., Ban, B., in [ISN97], pp329-337, Springer-Verlag, May 1997

[derrick] Formal Description Techniques for Object Management, Derrick, J., Linington, P.F., Thimpson, S.J., in [IM95], pp641-653, Chapman-Hall, 1995

[des403-B1011] WorldCom TMN Requirements Methodology Specification, WorldCom Engineering Standards, DES 403.1, Issue 1.01, Mar 1998

[dezen97] Proposal for an IN Switching State Model in an Integrated IN/B-ISDN Scenario, De Zen, G., Faglia, L., Hussmann, H., van der Vekens, A., in [ISN97], pp179-188, Springer-Verlag, May 1997

[dezen98] Accountable and Guaranteed Services in Internet, De Zen, G, Marsiglia, M., Ricagni, G., Vezzoli, L., in [ISN98], pp31-42, Springer-Verlag, May 1998

[dobson] The ORDIT Approach to Organisational Requirements, Dobson, J.E., Blyth, A.J.C., Chudg, J., Strens, R., Requirements Engineering: Social and Technical Issues, Academic Press, 1994

[EDOC97] Proceedings of the 1st International Workshop on Enterprise Object Distributed Computing,

[erdmann] Enterprise Modelling with FUNSOFT Nets, Erdmann, S., Wortmann, J., in [EDOC97], pp28-35, IEEE, 1997

[eriksson] The UML Toolkit, Eriksson, H., Penker, M., Wiley Computer Publishing, 1998

[etsi-na608] IN Intra Domain Management Requirements for CS-2, Draft TC-TR NA608-01, version7, 8/7/94, ETSI, Jul 1994

[fenton] Software Metrics, Fenton, E., Pfleeger, S.L., International Thompson Computer Press, 1997

[festor] MODE: A Development Environment for Managed Objects Based on Formal Methods, Festor, O., in [IM95], pp616-628, Chapman-Hall, 1995

[fink] Management Application Creation with DML, Fink, B., Dercks, H., Besting, P., in [IM95], pp629-640, Chapman-Hall, 1995

[fowler] UML Distilled - Applying the Standard Object Modeling Language, Fowler, M., Scott, K., 0-201-32563-2, Addison-Wesley, 1997

[furley] The BT operational support systems architecture, Furley, N., BT Technical Journal, vol. 15, No 1, January 1997, pp13-21, BT, 1997

[gagnon] A Security Architecture for TMN Inter-Domain Management, Gagnon, F., Maillot, D., Olnes, J., Hofseth, L., Sacks, L., in [ISN97], pp415-427, Springer-Verlag, May 1997

[galis] Towards Integrated Network Management for ATM and SDH Networks Supporting a Global Broadband Connectivity Management Service, Galis, A., Brianza, C., Leone, C., Salvatori, C., Gantenbein, D., Covaci, S., Mykoniatis, G., Karayannis, F., in [ISN97], pp303-314, Springer-Verlag, May 1997

[gamma] Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, E., Helm, R., Johnson, P., Vlissides, J., Addison-Wesley, 1995

[gaspoz] VPN on DCE: From Reference Configuration to Implementation, Gaspoz, J.P., Gbaguidi, C., Meinkohn, J., in [ISN95], pp249-260, Springer-Verlag, Oct 1995

[georgatsos] Technology Interoperation in ATM Networks: The REFORM System, Georgatsos, P., Makris, D., Griffin, D., Pavlou, G., Sartzetakis, S., T'Joens, Y., Ranc, D., IEEE Communications Magazine, vol. 37, no. 5, May 1999

[graham] Object Oriented Methods, Graham, I., 0-201-59371-8, Addison-Wesley, 1994

[graubmann] Engineering Modelling Concepts (DPE Architecture), Graubmann, P., Mercouroff, N.,, TINA Baseline Document TB_NA.005_2.0_94, TINA, 1994

[griffin95] A TMN System for VPC and Routing Management in ATM Networks, Griffin, D.P., Georgatsos, P., in [IM95], pp356-369, Chapman-Hall, 1995

[griffin96] Integrated Communications Management of Broadband Networks, Griffin, D. (ed), 960-524-006-8, Crete University Press, 1996

[griffin97] Implementing TMN-like Management Services in a TINA Compliant Architecture: A Case Study of Resource Configuration Management, Griffin, D., Pavlou, G., Tin, T., in [ISN97], pp263-274, Springer-Verlag, May 1997

[hall96] Management of Telecommunication Systems and Services: Modelling and Implementing TMN-based Multi-domain Management, Hall, J. (Ed), Lecture Notes in Computer Science 1116, Springer-Verlag, 1996

[hall98] Protocol Independent Information Modelling for a Peer-to-peer Configuration Interface, Hall, J., Best, M., Ferry, R., Fratini, S., Hunt, C., in [ISN98], pp193-204, Springer-Verlag, May 1998

[hegering] Integrated Network and System Management, Hegering, H.G., Abeck, S., 0-201-59377-7, Addison-Wesley, 1994

[hellemans99] Accounting Management in a TINA-Based Service and Network Environment, Hellemans, P., Redmond, C., Daenen, K., Lewis, D., in [ISN99], pp13-24, Springer-Verlag, Apr 1999

[herzog] From IN towards TINA - Potential Migration Steps, Herzog, U., Magedanz, T., in [ISN97], pp219-228, Springer-Verlag, May 1997

[hruby] Structuring Design Deliverable with UML, Hruby, P., in [UML98], pp251-260, OMG, Jun 1998

[ieee829] IEEE Standard for Software Test Documentation, IEE Standard 829, 1983

[IM95] Integrated Network Management IV: Proceedings of the 4th International Conference on Integrated Network Management, Santa Barbara, USA, Chapman-Hall, 1995

[IM97] Integrated Network Management V: Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management, San Diego, USA, San Diego, USA, Chapman-Hall, 1997

[ISN94] Proceedings of the 2nd International Conference on Intelligence in Services and Networks, Aachen, Germany, Springer-Verlag, 1994

[ISN95] Proceedings of the 3rd International Conference on Intelligence in Services and Networks, Heraklion, Greece, Springer-Verlag, 1995

[ISN97] Proceedings of the 4th International Conference on Intelligence in Services and Networks, Cernobbio, Italy, Springer-Verlag, 1997

[ISN98] Proceedings of the 5th International Conference on Intelligence in Services and Networks, Antwerp, Belgium, Springer-Verlag, 1998

[ISN99] Proceedings of the 6th International Conference on Intelligence in Services and Networks, Barcelona, Spain, Springer-Verlag, 1999

[itu-odl] ITU - Object Definition Language (ITU-ODL), ITU-T, Jan 1998

[jacobsen92] Object-Oriented Software Engineering, Jacobsen, I., Chisterson, M., Jonsson, P., Overgaard, G., 0-201-54435-0, Addison-Wesley, 1992

[jacobsen97] Software Reuse - Architecture, Process and Organisation for Business Success, Jacobsen, I., Griss, M., Jonsson, P., 0-201-92476-5, Addison-Wesley, 1997

[kande] Applying UML to Design an Intrer-Domain Service Management Application, Kande, M.M., Mazahaer, S., Prajat, O., Sacks, L., Wittig, M., in [UML98], pp173-182, OMG, Jun 1998

[karlsson] Software Reuse: A Holistic Approach, Karlsson, E.A., 0=471-95819-0, Wiley, 1996

[keil] The Mobilise enterprise model: foundation and application, Keil, K., Niebert, N., Kugler, H.J., Proceeding of the RACE IS&N Conference, 1992

[kindel] COM: What Makes It Work - Black Box Encapsulation through Multiple, Immutable Interface, Kindel, C., in [EDOC97], pp68-77, IEEE, 1997

[kitson] CORBA and TINA: The Architectural Relatioship, Kitson, B., Proceeding of TINA'95, Melbourne, Australia, pp371-386, TINA-C, Feb 1995

[kivisto] Considerations of and Suggestions for a UML-Specific Process Model, Kivisto, K., in [UML98], pp261-271, OMG, Jun 1998

[korthaus] BOOSTER* Process: A Software Development Process Model Integrating Business Object Technology and UML, Korthaus, A., Kuhlins, S., in [UML98], pp205-214, OMG, Jun 1998

[leakey] Some technical aspects of regulation, Leakey, D., Telecommunications, 26-29 March 1995, Conference Publication No. 404, pp278-281, IEE, 1995

[lewis94] A Broadband Testbed for the Investigation of Multimedia Services and Teleservice Management, Lewis, D., Kirstein, P., Proceedings of the 3rd International Conference on Broadband Islands, Hamburg, Germany, April 1994

[lewis95a] Experiences in Multi-Domain Management System Development, Lewis, D., O'Connell, S., Donnelly, W., Bjerring, L., in [IM95], pp494-505, Chapman-Hall, 1995

[lewis95b] Experiences in Multi-Domain Management Service Development, Lewis, D., Tiropanis, T., Bjerring, L.H., Hall, J., in [ISN95], pp174-184, Springer-Verlag, Oct 1995

[lewis97] Inter-Domain Integration of Services and Service Management, Lewis, D., Tiropanis, T., Redmond, C., Wade, V., McEwan, A., Bracht, R., in [ISN97], pp283-292, Springer-Verlag, May 1997

[lewis98a] Integrating TINA into an Internet-Based Service Market, Lewis, D., Tiropanis, T., in [ISN98], pp185-192, Springer-Verlag, May 1998

[lewis99a] A Development Framework for Open Management Systems, Lewis, D., Journal of Interoprable Communication Networks, vol. 2/1, pp11-30, Mar 1999

[lewis99b] Modelling Management Components for Reuse Using UML, Lewis, D., Malbon, C., DaCruz, A., in [ISN99], pp210-222, Springer-Verlag, Apr 1999

[lewis99c] A Review of Approaches to Developing Service Management Systems, Lewis, D., to appear in Journal of Systems and Network Management, 1999

[lewis99d] The Development of Integrated Inter and Intra Domain Management Services, Lewis, D., Wade, V., Bracht, R., Integrated Network Management VI: Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, Boston, USA, pp279-292, Addison-Wesley, May 1999

[lewis99e] The Component-based Integration of Customer Subscription Management with Network Planning and Network Provisioning, Lewis, D., Palou, G., Malbon, C., Stathopoulos, C., Villoldo, J.E., to be published in the proceedings of Proceedings of the 10th IFIP/IEEE International Workshop on Distrubted Systems Operations and Management , 1999

[lodge] Alignment of the TOSCA and SCREEN Approaches to Service Creation, Lodge, F., Kimbler, K., Hubert, M., in [ISN99], pp277-290, Springer-Verlag, Apr 1999

[lucidi] Development of TINA-like Systems: The DOLMAN Methodology, Lucidi, F., Idzenga, H., Batistatos, S., in [ISN98], pp379-392, Springer-Verlag, May 1998

[m3000] Overview of TMN Recommendations, ITU-T Recommendation M.3000, 1995

[m3010] Principles for a Telecommunications Management Network, ITU-T Recommendation M.3010, 1996

[m301x] Definitions of Principles and Concepts for a Telecommunications Management Network, ITU-T Draft Recommendation M.301x, Jun 1997

[m301y] Considerations for a Telecommunications Management Network, ITU-T Draft Recommendation M.301y, Aug 1996

[m3020-95] TMN Interface Specification Methodology, ITU-T Recommendation M.3020, 1995

[m3020-99] TMN Interface Specification Methodology, ITU-T Draft Recommendation M.3020 Revision, Jul 1999

[m3100] Generic Network Information Model, ITU-T Recommendation M.3100, 1992

[m3200] TMN Management Services: Overview, ITU-T Recommendation M3200, 1992

[m3400] TMN Management Functions, ITU-T Recommendation M.3400, 1997

[magedanz] Modeling IN-based service control capabilities as part of TMN-based service management, Magedanz, T., in [IM95], pp387-397, Chapman-Hall, 1995

[marcus] Icaros, Alice and the OSF DME, Marcus, J.S., in [IM95], pp83-92, Chapman-Hall, 1995

[martin] Adopting Object Oriented Analysis for Telecommunications Systems Development, Martin, D., in [ISN97], pp117-128, Springer-Verlag, May 1997

[maston] Using the World Wide Web and Java for Network Service Management, Maston, M.C., in [IM97], pp71-84, Chapman-Hall, 1997

[may] The Relationship Between IOs and COs in VPN Charging Management, May, J., Maia, A., in [ISN95], pp185-199, Springer-Verlag, Oct 1995

[mccarthy] Exploiting the Power of OSI Management for the Control of SNMP Capable Resources Using Generic Application Level Gateways, McCarthy, K., Pavlou, G., Bhatti, S., Neuman do Souza, J., in [IM95], pp440-453, Chapman-Hall, 1995

[mcleod] Extending UML for Enterprise and Business Process Modelling, McLeod, G., in [UML98], pp195-204, OMG, Jun 1998

[mercouroff95] TINA Object Definition Language (TINA-ODL) Manual, Mercouroff, N., Kitson, B. (eds), TINA Baseline Document, TR_NM.002_1.3_95, TINA-C, Jun 1994

[mercouroff97] TINA Computational Modelling Concepts and Object Definition Language, Mercouroff, N., Parhar, A., in [ISN97], pp15-24, Springer-Verlag, May 1997

[meszaros] Distributed Objects in Telecommunications, Meszaro, G., in [EDOC97], pp149-159, IEEE, 1997

[milsted] OMT Object Modelling of Telecommunications Services, Milsted, K., in [ISN95], pp369-379, Springer-Verlag, Oct 1995

[monton] Maintaining Integrity in the Context of Intelligent Networks and Services, Monton, V., Ward, K., Wilby, M., in [ISN97], pp427-436, Springer-Verlag, May 1997

[morris] An SDL based Realisation of an IN Service Development Environment, Morris, C., Nelso, J., in [ISN95], pp292-308, Springer-Verlag, Oct 1995

[mowbray] CORBA Design Patterns, Mowbray, T., Malveau, R., 0-471-15882-8, Wiley Computer Publishing, 1997

[mulder] TINA Business Model and Reference Points, v4.0, Mulder, H. (ed), TINA baseline document, TINA-C, May 1997

[natarajan] Computational Modelling Concepts, Natarajan, N., Dupuy, F., Singer, N., Christensen, H., TINA Baseline Document, TB_A2.HC.012_1.2_94, TINA, 1994

[nesbitt] The EURESCOM P610 Project: Providing a Framework, Architecture and Methodology for Multimedia Service Management, Nesbitt, F., Counihan, T., Hickie, J., in [ISN98], pp73-88, Springer-Verlag, May 1998

[nielsen] Development of Telecommunications Management Systems Using OO Methods and CASE Tool Support, Nielsen, P.S., Lonvig, B., in [ISN94], pp407-418, Springer-Verlag, Sep 1994

[nmf-025] The "Ensemble" Concept and Format, NMF 025, Issue 1.0, NMF, Morristown, 1992

[nmf-504] SMART Ordering - SP to SP Interface Business Agreement, NMF 504, Issue 1.0, TMF, Sep 1997

[nmf-gb901] A Service Management Business Process Model, GB901, NMF, Morristown, 1995

[nmf-gb908] A Network Management Detailed Operations Map, NMF, Morristown, 1998

[nmf-gb909] SMART TMN Technology Integration Map, GB 909, Issues 1.1, TMF, Oct 1998

[nmf-gb910] NMF Telecoms Operation Map: A high-level view of end-to-end service fulfilment, service assurance and billing, NMF, Morristown, 1998

[noam] Beyond liberalisation: From the network of networks to the systems of systems, Noam, E., Telecommunications Policy, v18(4), pp286-294, Butterworth-Heinemann, 1994

[olsen95] Using SDL for Targeting Service to CORBA, Olsen, A., Norbaek, B.B., in [ISN95], pp334-346, Springer-Verlag, Oct 1995

[olsen99] The Pros and Cons of Using SDL for Creation of Distributed Services, Olsen, A., Demany, D., Cardoso, E., Lodge, F., Kolberg, M., Bjorkander, M., Sinnott, R., in [ISN99], pp342-354, Springer-Verlag, Apr 1999

[oma-cos] Common Object Services, vol. 1 and 2, OMG, 1995

[omg/96-01-04] Multiple Interfaces and Composition RFP, OMG, Jan 1996

[oppenheim] Questionnaire Design, Interviewing and Attitude Measurement, Oppenhiem, A.N., 1-85567-0437, Pinter Publishers, 1992

[orbos/99-02-05] CORBA Components: Joint Revised Submission, OMG TC Document orbos/99-02-05, Mar 1999

[orbos/99-04-11] CORBA Management: ORB Instrumentation, v1.0, orbos/99-04-11, OMG, Apr 1999

[orfali] Client/Server Programming with Java and CORBA, 2nd ed, Orfli, R., Harkey, D., 0-471-24578-X, Wiley Computer Publishing, 1998

[p414-d2] Project P414, TMN Guidelines,: Deliverable 2, TMN Design Case Study Report - Overview, EURESCOM, Jul 1996

[p414-d3] Project P414, TMN Guidelines,: Deliverable 3, TMN Guidelines, EURESCOM, Aug 1996

[p610-d1] Report of state of the art on current activities on management framework, methodologies and case study service selection criteria for management of multimedia service, Ferrari, L., EURESCOM, Feb 1997

[p610-d2] Management Framework and Methodology, de la Fuente, L., Gallego, J., Llamas, P. (eds), EURESCOM, 1997

[pavlou94] High-Level Access APIs in the OSIMIS TMN Platform: Harnessing and Hiding, Pavlou, G., Tin, T., Carr, A., in [ISN94], pp181-191, Springer-Verlag, 1994

[pavlou95a] Issues in the integration of IN and TMN, Pavlou, G., Griffin, D., Bringing Telecommunication Services to the People, Proceedings of the 3rd International conference on Intelligence in Broadband Services and Networks, Springer-Verlag, 1995

[pavlou95b] The OSIMIS Platform: Making OSI Management Simple, Pavlou, G., McCarthy, K., Bhatti, S., Knight, G., in [IM95], pp480-493, Chapman-Hall, 1995

[pope] The CORBA Reference Guide, Pope, A., 0-201-63386-8, Addison-Wesley, 1998

[potonniee] Implementing TMN using CORBA Object Distribution, Potoniee, O., Hauw, L.H., Ranc, D., Bardout, Y., Canela, Z., Proceedings of the International Conference on Management of Multimedia Networks and Services, Montreal, Canada, pp83-94, Chapman-Hall, 1997

[prnjat] Integrity Methodology for Interoperability Environments, Prnjay, O., Sacks, L., IEEE Communications Magazine, vol. 37, no. 5, pp126-132, 1999

[putter] Towards Policy Driven Systems Management, Putte, P., Bishop, J., Roos, J., in [IM95], pp69-80, Chapman-Hall, 1995

[q1200] Q-series Intelligent Network Recommendation Structure, ITU-T Recommendation Q.1200, 1993

[rahkila] Experiences on Building Distributed Computing Platform Prototype for Telecom Network and Service Management, Rahkila, S., Stenberg, S., in [IM97], pp127-138, Chapman-Hall, May 1997

[rasmussen] A CORBA to CMIP Gateway: A Marriage of Management Technologies, Rasmussen, S., Baumer, C., in [ISN98], pp477-492, Springer-Verlag, May 1998

[rumbaugh] Object-Oriented Modelling and Design, Rumbaugh, J., Blaha, W., Premerlani, W., Eddy, F., Lorensen, W., 0-13-630054-5, Prentice-Hall, 1991

[salleros] TINA-C Service Design Guidelines, Salleros, J., TINA Report TP_JS_001_0.1_95, TINA, 1995

[saydam] Object-Oriented Design of a VPN Bandwidth Management System, Saydam, T., Gaspoz, J.P., in [IM95], pp344-355, Chapman-Hall, 1995

[schieferdecker] Conformance Testing of TINA Service Components - The TTCN/CORBA Gateway, Schieferdcker, I., Li, M., Hoffmann, A., in [ISN98], pp393-408, Springer-Verlag, May 1998

[schoo] Modularization of TINA Reference Points for Information Networking, Schoo, P., Egelhaaf, C., Eckardt, T., Agoulmine, N., Tschichholz, M., in [ISN99], pp443-445, Springer-Verlag, Apr 1999

[shrewsbury] Part II: Technology Direction Statement - Building Block Requirements, Shrewsbury, K. (ed), TMF, 1998

[sloman] Domains: A Framework for Structuring Management Policy, Sloman, M., Twidle, K., Network and Distributed Systems Management, pp433-453, Addison-Wesley, 1994

[soukouti] Join Inter Domain Management: CORBA, CMIP and SNMP, Soukouti, N., Hollberg, U., in [IM97], pp153-164, Chapman-Hall, May 1997

[stallings] SNMP, SNMPv2 and CMIP: The Practical Guide to Network Management Standards, Stallings, W., 0-201-63331-0, Addison-Wesley, 1993

[stathopoulos] Handling the Distribution of Information in the TMN, Stathopoulos, C., Griffin, D., Sartzetakis, S., in [IM95], pp398-411, Chapman-Hall, 1995

[strens] Responsibility Modelling as a Technique for Organisational Requirements Definition, Strens, R., Dobson, J., Intelligent Systems Engineering, vol. 3, no. 1, pp20-26, 1994

[strick94] Specifying Pan-European Management Systems, Strick, L., Meinkohm, J., in [ISN94], pp467-478, Springer-Verlag, Sep 1994

[strick96] Development of IBC Service Management Services, Strick, L., Wittig, M., Paschke, S., Meinkohn, J., Proceedings of the IFIP/IEEE Network Operations Management Symposium, ppKyoto, Japan, Apr 1996

[stringer] CORBA-based Telecommunication Network Management Systems, Stringer, D., Rutt, T. (eds), OMG White Paper, May 1996

[sullivan] A Comparison of the PRISM and ONMI-Point Methodologies for the Specification of Management Systems, Sullivan, D., McLaughlin, P., in [ISN94], pp553-563, Springer-Verlag, 1994

[tina-nra] TINA Network Resource Architecture, v3.0, TINA Baseline Document TB_FS.001_3.0_97, TINA-C, 1997

[tiropanis97] A Service Engineering Approach to Inter-Domain TMN System Developer, Tiropanis, T., Lewis, D., Richter, A., Shi, R., in [IM97], pp165-177, Chapman-Hall, May 1997

[tiropanis98] Offering Role Mobility in a TINA Environment, Tiropanis, T., in [ISN98], pp89-100, Springer-Verlag, May 1998

[UML98] Proceedings of UML'98 Conference, Mulhouse, France, OMG, 1998

[valiant] Review of software tools and methods used in operational support systems developments, Valiant, S.C., BT Technical Journal, vol. 15, No 1, January 1997, pp147-150, BT, 1997

[varley] User Administration and Accounting, Varley, B., Network and Distributed Systems Management, pp381-402, Addison-Wesley, 1994

[vincent] Modeling/Design Methodology and Template, Draft 4, Vincent, A, Hall, C., TMF, Oct 1997

[vlissides] Pattern Languages of Program Design 2, Vlissides, J., Coplien, J., Kerth, N, 0-201-89527-7, Addisson-Wesley, 1996

[wade97] A Methodology for Developing Integrated Multi-domain Service Management Systems, Wade, V., Lewis, D., Sheppard, M., Tschichholz, M., Hall, J., in [ISN97], pp245-244, Springer-Verlag, May 1997

[wade98] A Design Process for the Development of Multi-Domain Service Management Systems, Wade, W., Lewis, D., Donnelly, W., Ranc, D., Karatzas, N., Guidelines for ATM Deployment and Interoperability, pp88-103, Baltzer Science Publishers, 1998

[wade99] Three Keys to Developing and Integrating Telecommunications Service Management Systems, Wade, V., Lewis, D., IEEE Communications Magazine, vol. 37, no. 5, pp140-146, 1999

[wies] Using a Classification of Management Policies for Policy Specification and Policy Transformation, Wies, R., in [IM95], pp44-56, Chapman-Hall, 1995

[x407] Message Handling Systems: Abstract Service Definition Conventions, ITU-T Recommendation X.407/ ISO/IEC International Standard 10021-3, 1998

[x700] Management framework for Open Systems Interconnection (OSI) for CCITT applications, CCITT Recommendation X.700, 1992

[x708] Information Technology - Open Systems Interconnection - Open Distributed Management Architecture, ITU-T Draft Recommendation X.708, June 1996

[x711] Information Technology - Open Systems Interconnection - Common Management Information Protocol Specification, ITU-T Recommendation X.711 - ISO/IEC 10165-1, 1993

[x722] Information Technology - Open Systems Interconnection -Structure of management information: Guidelines for the Definition of Managed Objects, ITU-T Recommendation X.722, 1992

[x725] Information Technology- Open Systems Interconnection- Structure of Management Information- Part 7: General Relationship Model, ITU-T Recommendation X.725/ ISO/IEC Draft International Standard 10165-7, ITU-T, 1994

[x734] Information Technology - Open Systems Interconnection -Systems management: Event Report Management Function, ITU-T Recommendation X.734, 1993

[x741] Information Technology - Open Systems Interconnection -Systems management: Objects and Attributes for Access Control, ITU-T Recommendation X.741, 1995

[x750] Information Technology - Open Systems Interconnection -Systems management: Management Knowledge Management Function, ITU-T Recommendation X.750, 1996

[x901] Open Distributed Processing- Reference Model: Part 1: Overview and Guide to Use, ITU-T Recommendation X.901/ ISO/IEC International Standard 10746-1, 1995

[x902] Open Distributed Processing - Reference Model: Part 2: Foundations, ITU-T Recommendation X.902/ ISO/IEC International Standard 10746-2, 1995

[x903] Open Distributed Processing - Reference Model: Part 3: Architecture, ITU-T Recommendation X.903/ ISO/IEC International Standard 10746-3, 1995

[x904] Open Distributed Processing- Reference Mode: Part 4: Architectural Semantics, ITU-T Draft Recommendation X.904/ISO Draft International Standard 10746-4, 1994

[xml] Extensible Markup Language (XML) 1.0, W3C Recommendation: REC-xml-19980210, World Wide Web Consortium, Feb 1998

[yavatkar] A Framework for Policy-based Admission Control, Yakatkar, R., Pendarakis, D., Guerin, R., draft-ietf-rap-framework-01.txt, IETF, May 1998

[z100] Specification and Description Language, ITU-T Recommendation Z.100, Addendum 1, Oct 1996

[zeisler] A Framework for System and Network Management Ensembles, Zeisler, E.D., Folts, H.C., in [IM95], pp602-614, Chapman-Hall, 1995

[zelkowitz] Experimental Models for Validating Technology, Zelkowitz, M., Dolores, W., Computer, pp23-31, IEEE, May 1998

9. Appendix 1

This appendix summarises the results of the responses to the questionnaires conducted on the developers involved in Case Studies 4 and 5. The responses presented are those relating to the assessment of how useful the various techniques were found to be measured on a scale of:

Essential =5, Mostly Useful =4, Generally Useful =3, Partially Useful =2, Not Useful =1.

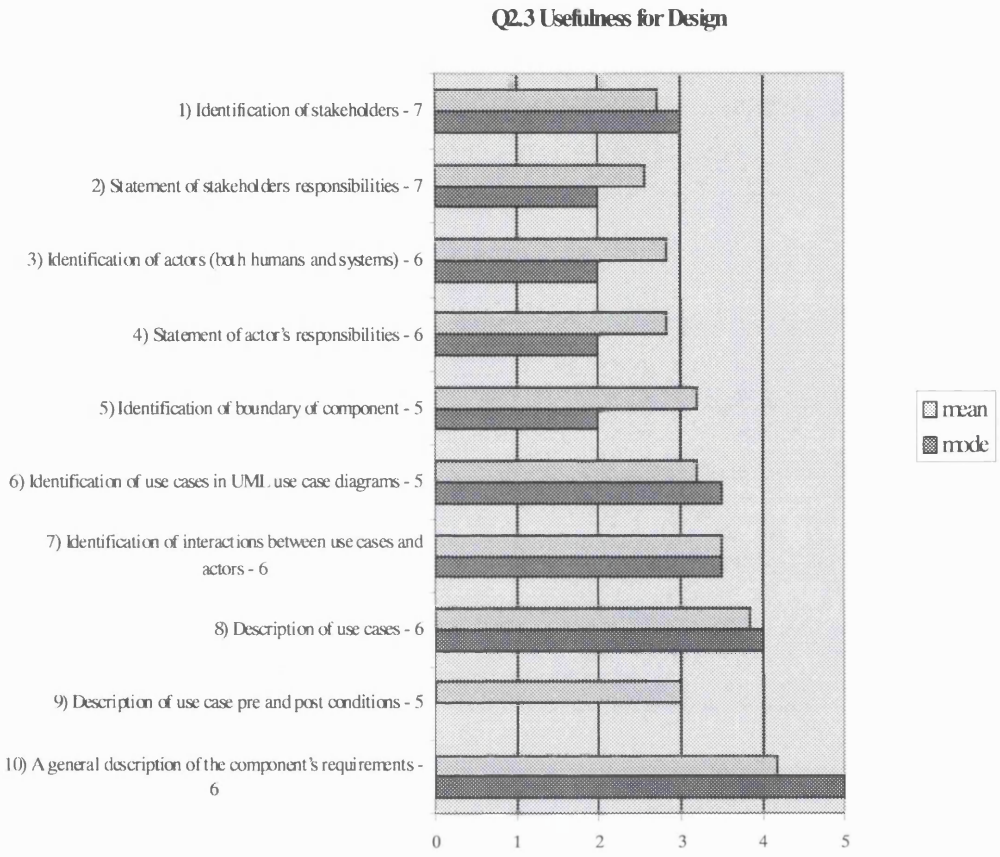
The responses to introductory, filter or personal information questions are not presented. The mode response for each technique address in each question is given. Though this not an ordinal scale the mean is also presented to give a further indication of the spread of responses. In addition, if two adjacent values contained the same mode total the mode is presented as the median of those two values. If the mode was present in two unadjacent categories or in more then two categories then no mode value is presented. For Case Study 3 questions were also asked in each category in order to assess the problems encountered at each stage in development due to concurrent or previous activity output. However, due to poor phrasing of the question the results were confounded and are not presented here.

9.1 Case Study 4 Response Summary

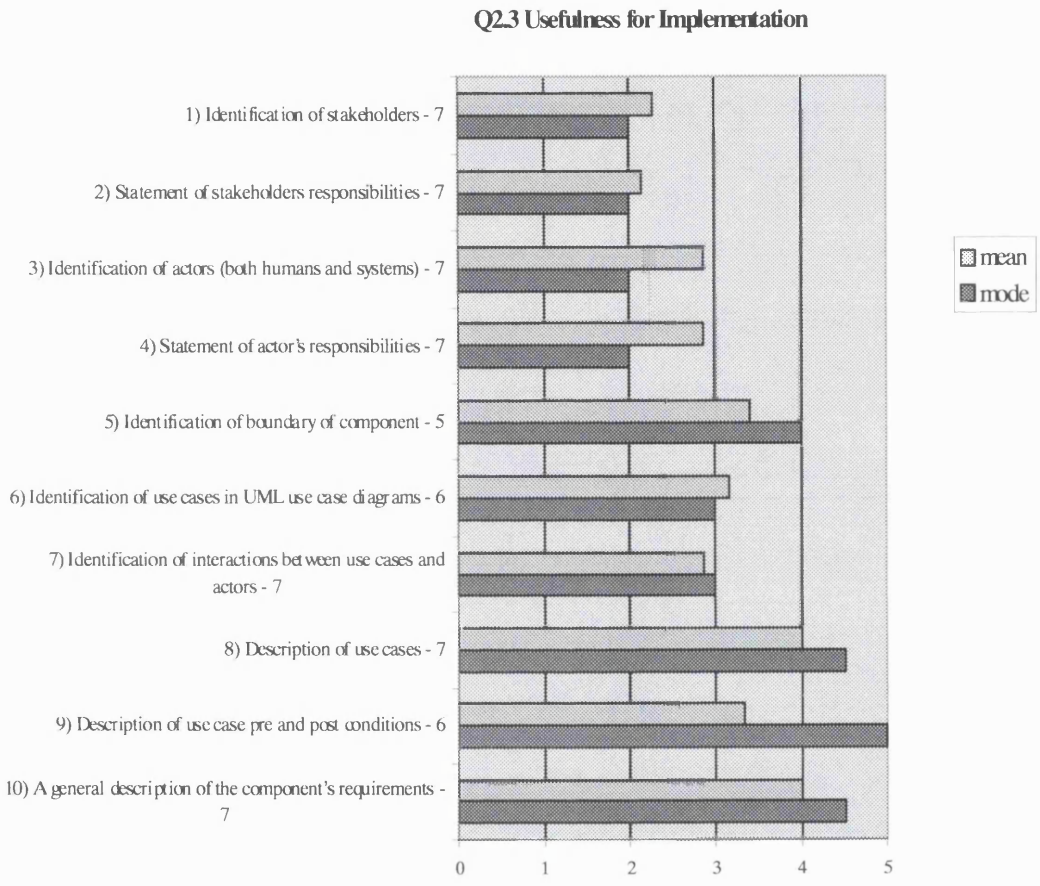
9.1.1 Responses from Component Developers

Q2.3) Please indicate **how useful** the following parts of the **analysis specification** were to you in the further development of the component. Please indicate the usefulness to you of each part of the analysis specification during both the design and implementation phases of your component.

Responses for design phase:

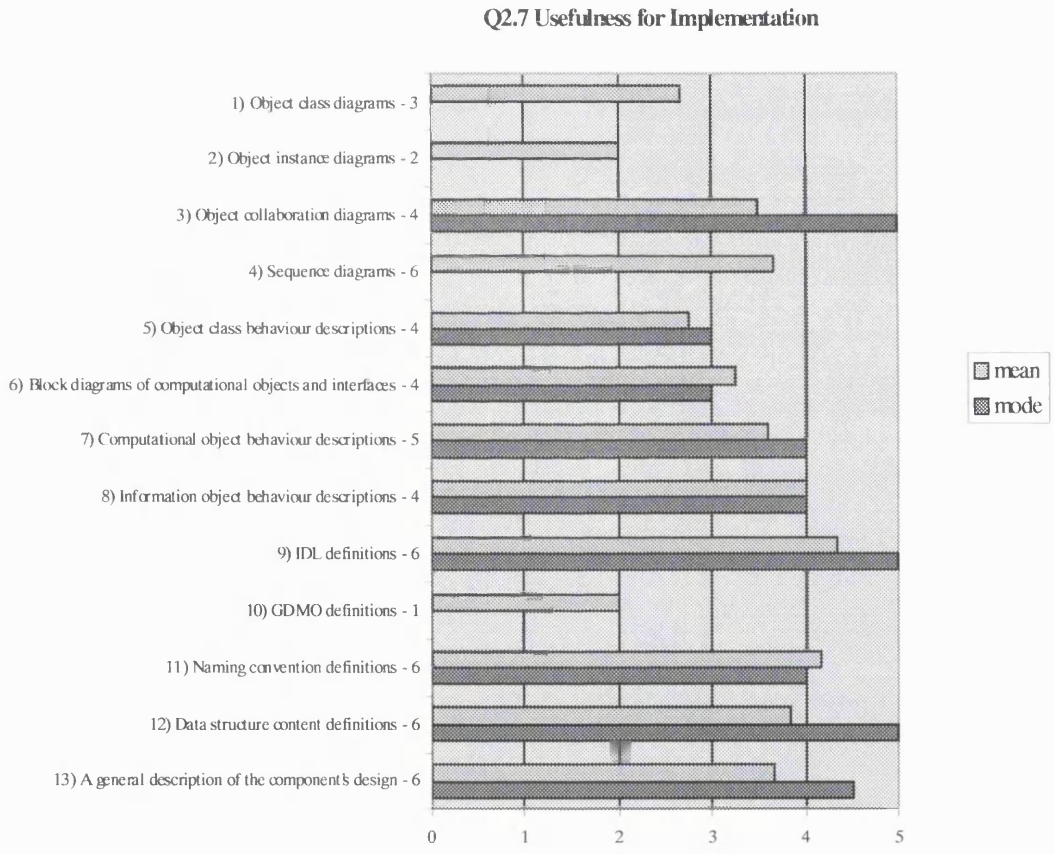


Responses for implementation phase:

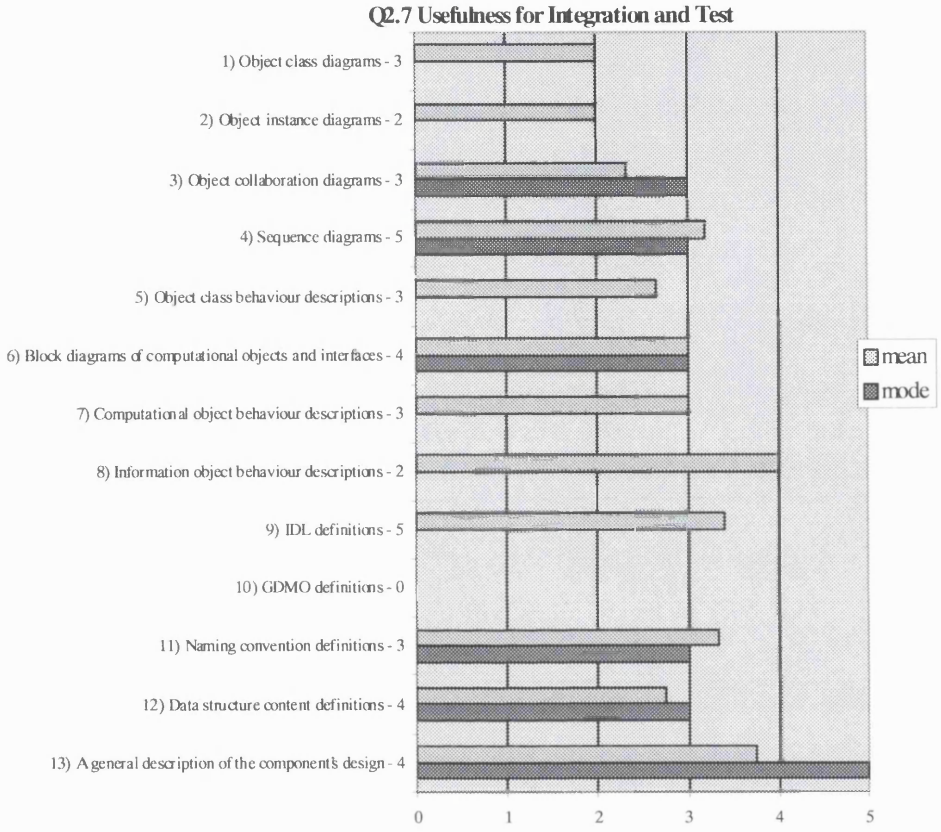


Q2.7) Please state **how useful** you found the parts of the **design specifications** that you used. Please indicate the usefulness to you of each part of the design specification during both the implementation and testing/integration phases of your component.

Responses for implementation phase:



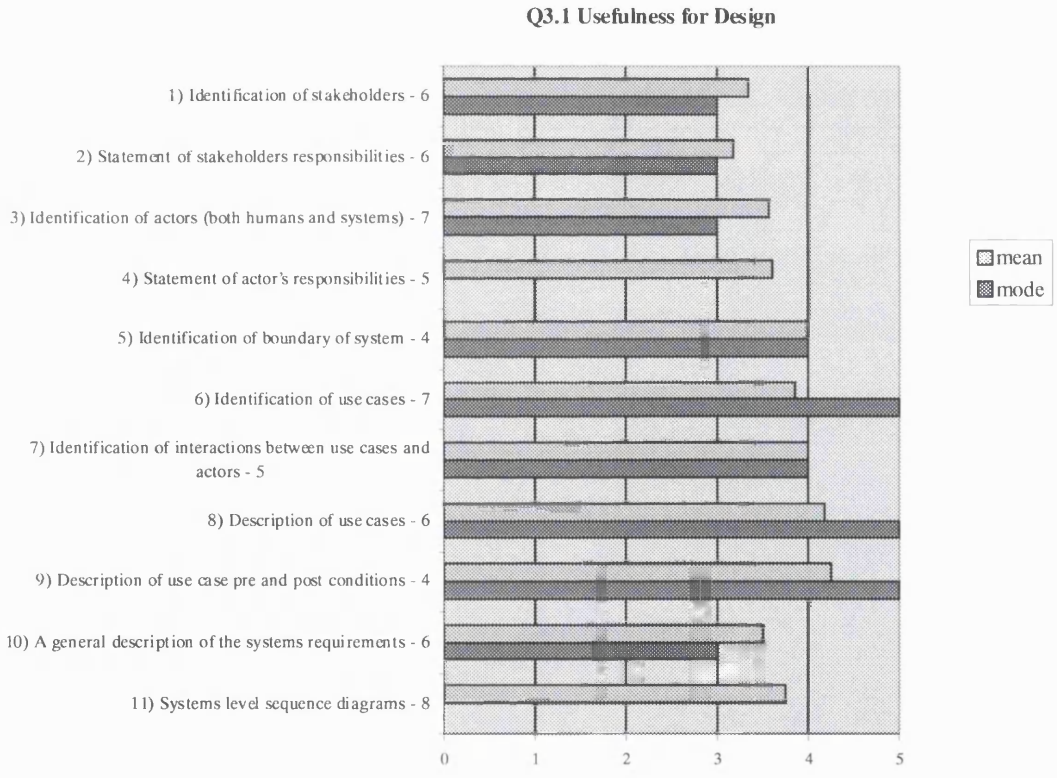
Responses for testing/integration phases:



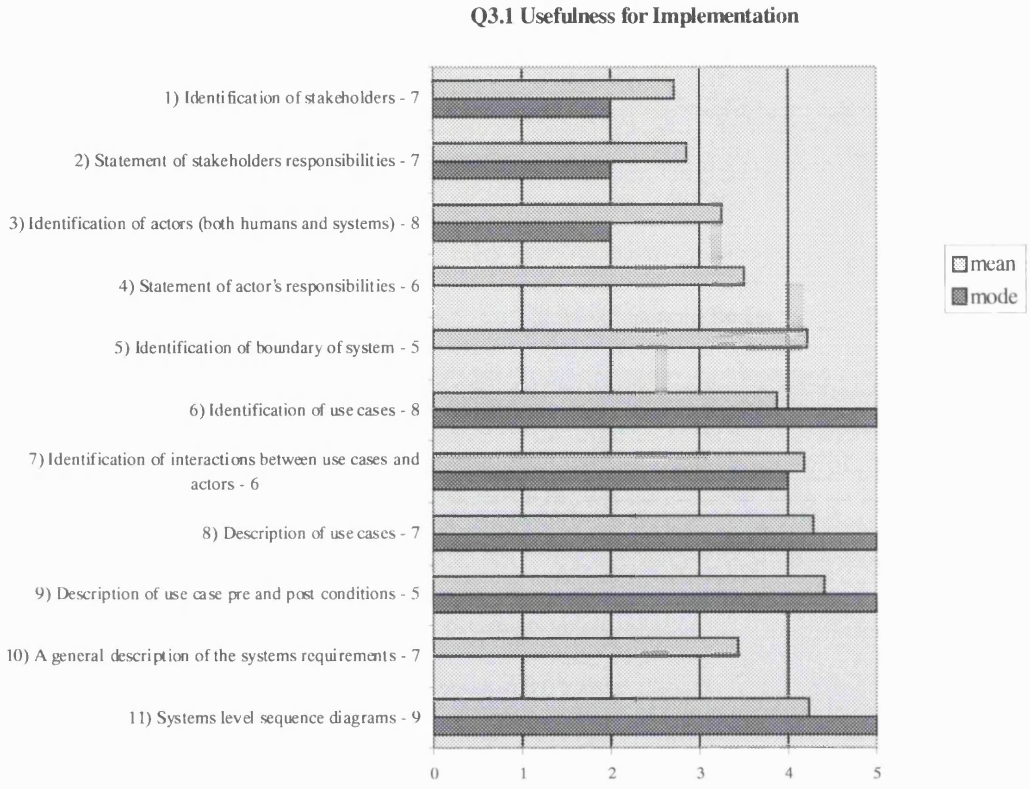
9.1.2 Responses from System Developers

Q3.1) Please state **how useful** you found the different parts of **multi-domain analysis specifications** of the trials containing the systems you worked on. Please indicate the usefulness to you of each part of the multi-domain analysis specifications during both the design and implementation phases of your system.

Responses for design phase:

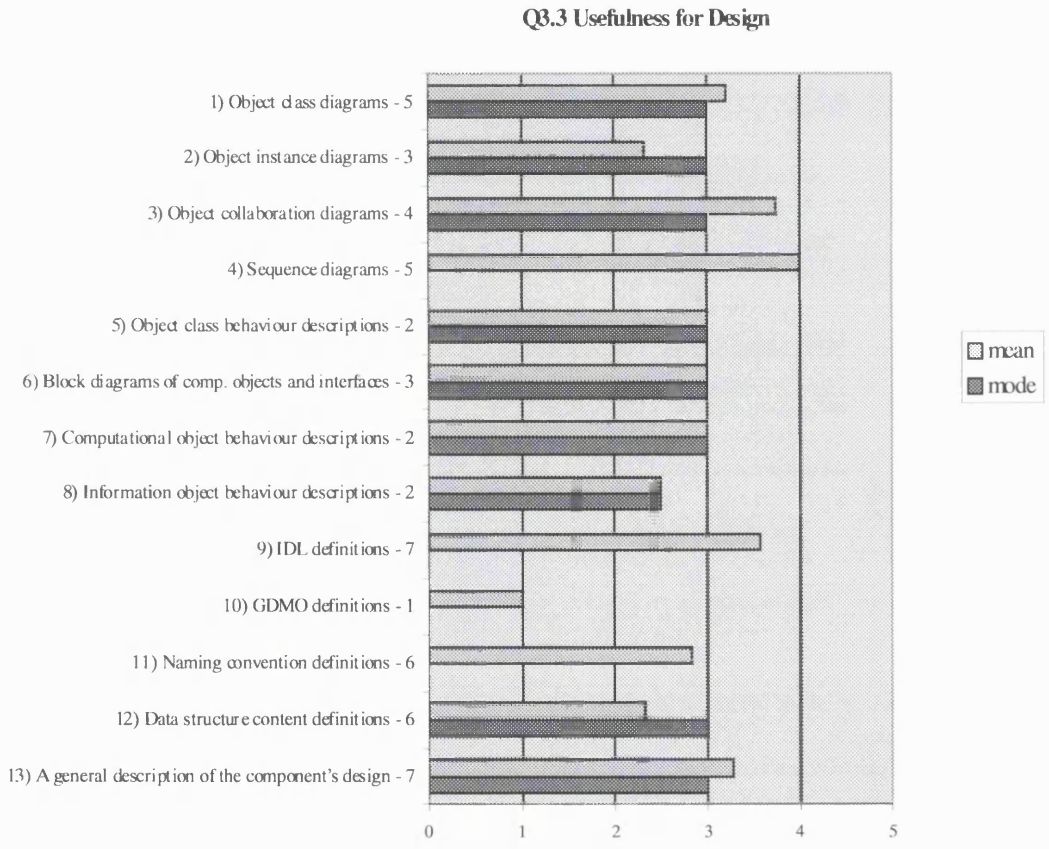


Responses for implementation phase:

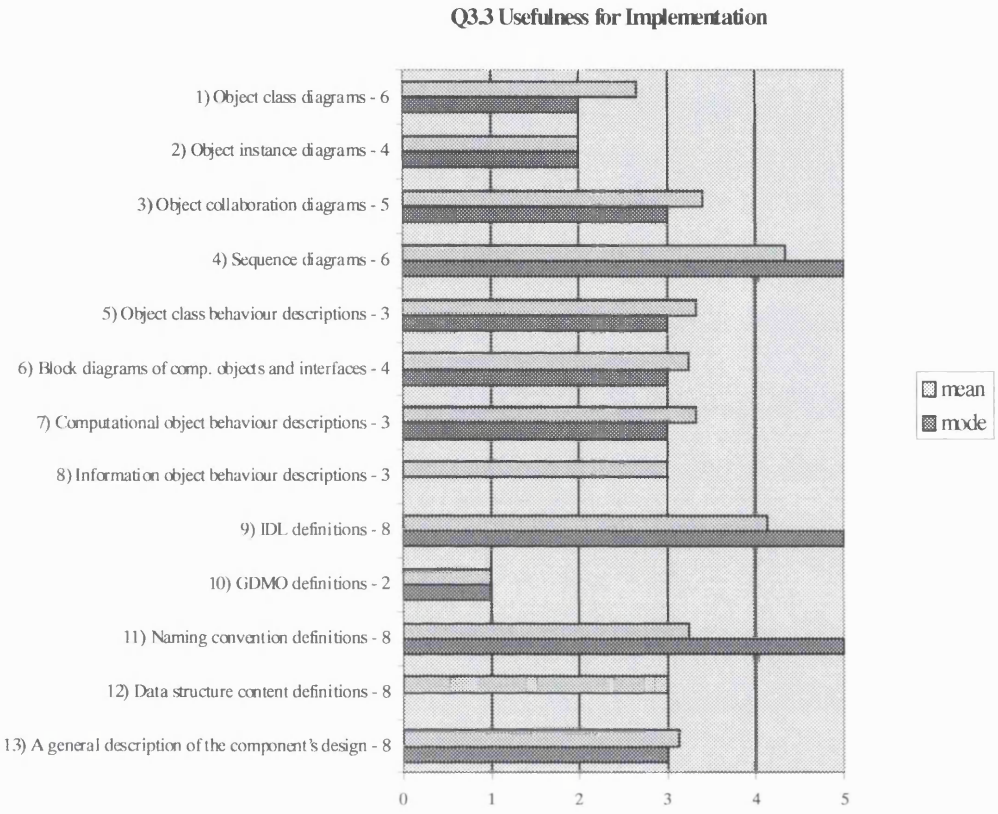


Q3.3) Please state **how useful** you found the **design specifications of other systems** with which your system had to inter-operate. Please indicate the usefulness to you of each part of the design specification during the design, implementation and testing/integration phases of your system.

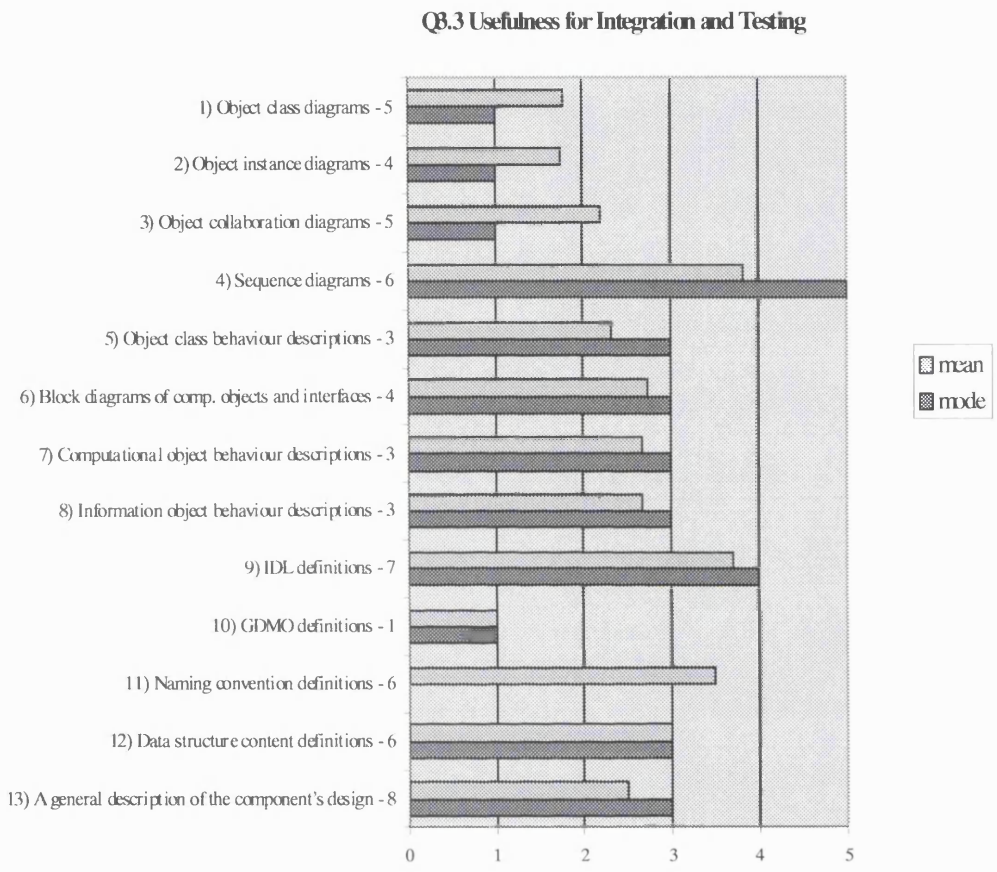
Responses for design phase:



Responses for implementation phase:

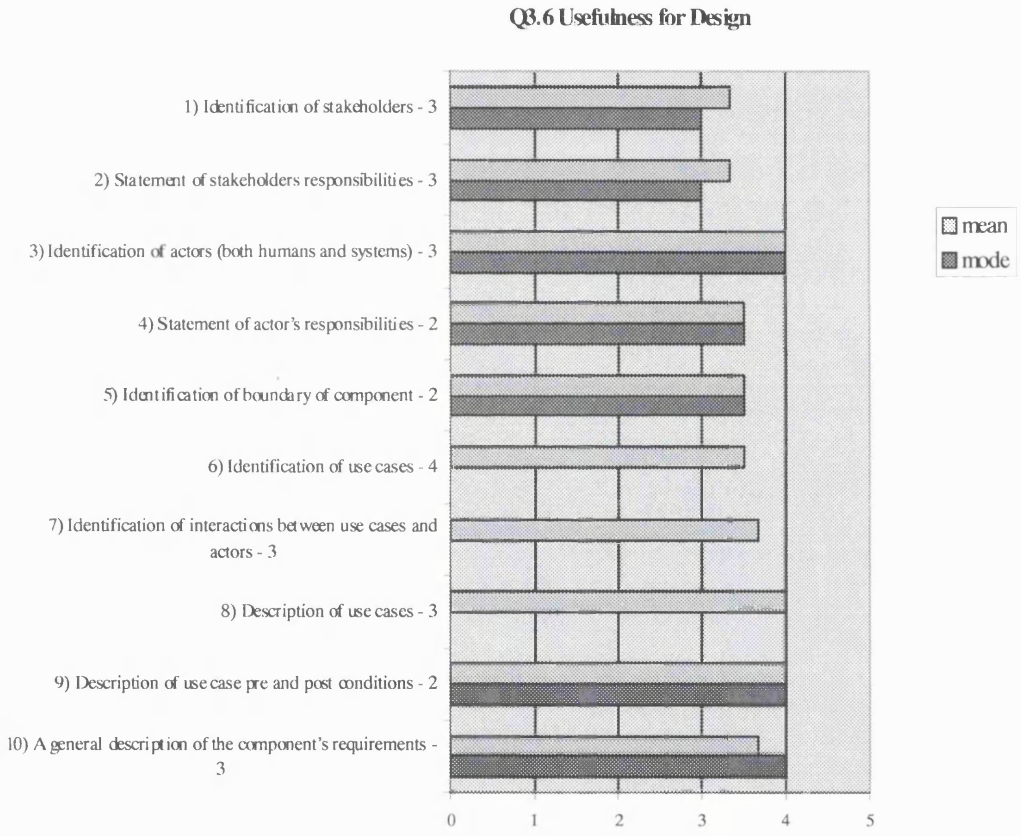


Responses for testing/integration phases:

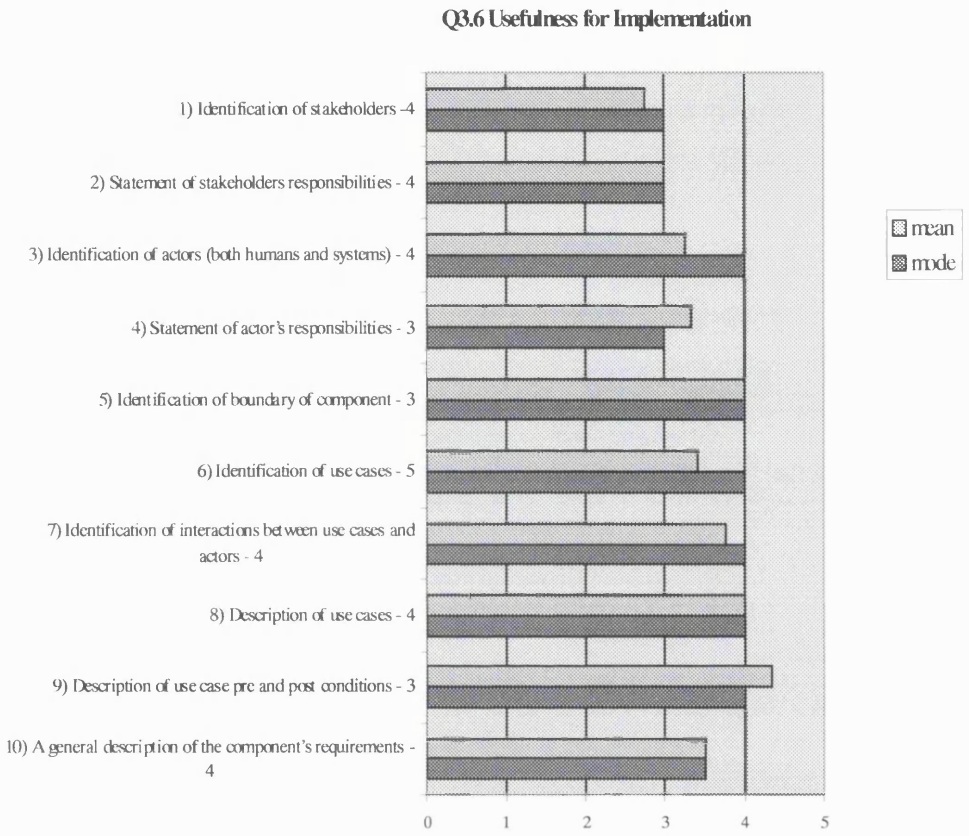


Q3.6) Please indicate **how useful** the following parts of the **components' analysis specifications** were to you in the development of the system. Please indicate the usefulness to you of each part of the components' analysis specifications during both the design and implementation phases of your system.

Responses for design phase:

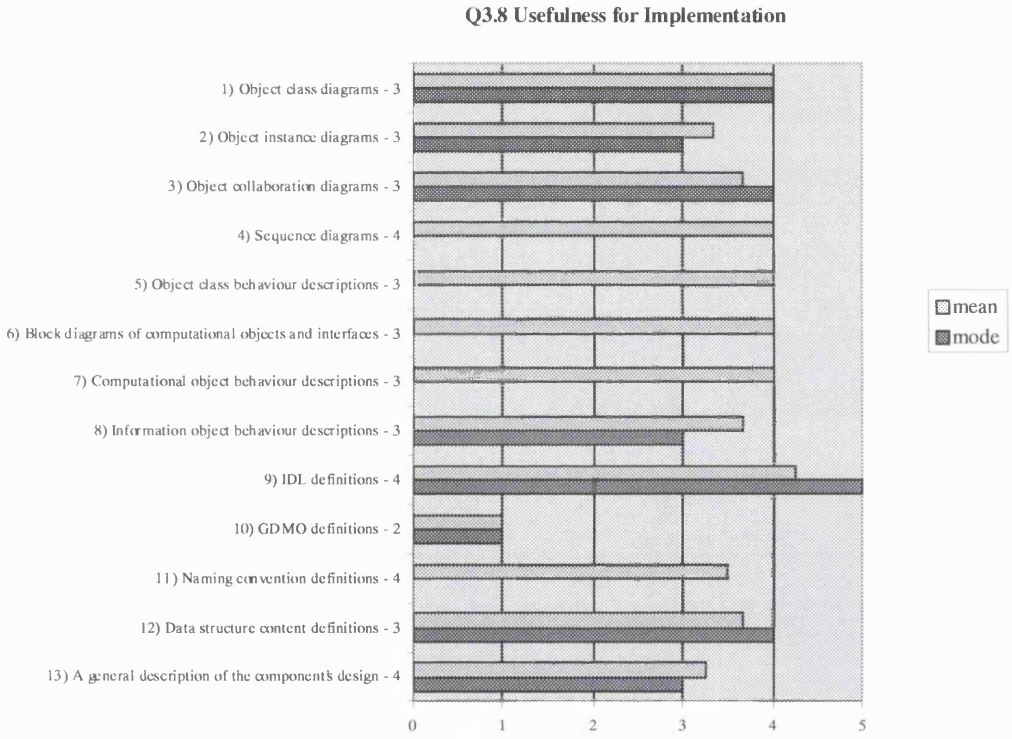


Responses for implementation phase:

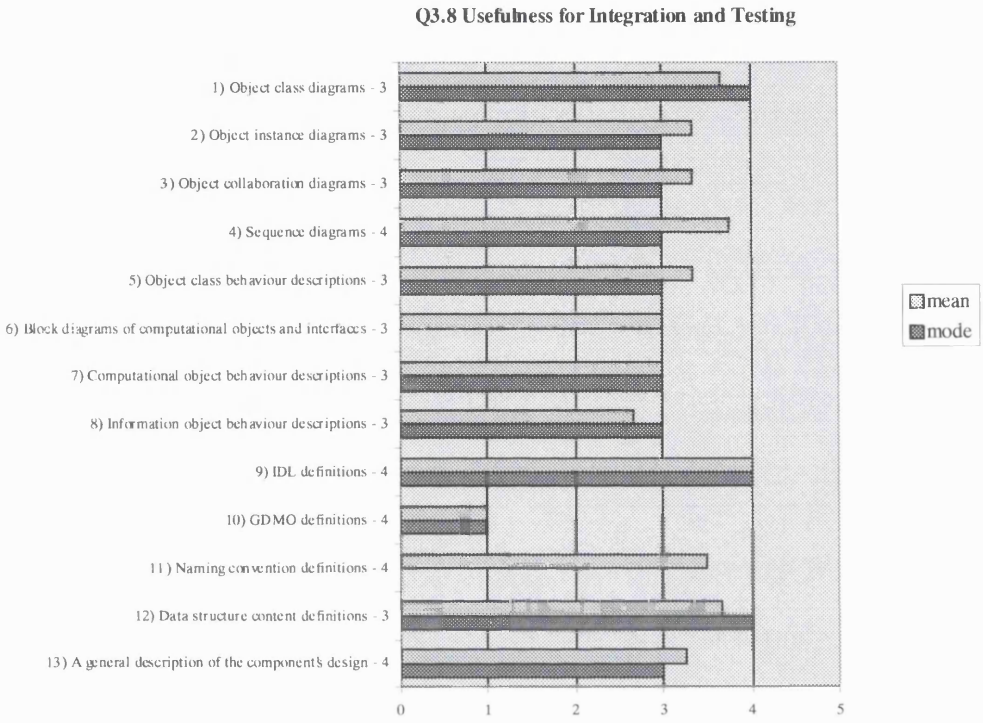


Q3.8) Please indicate **how useful** did you find the parts of the **components' design specifications** you used? Please indicate the usefulness to you of each part of the components' design specifications during both the implementation and testing/integration phases of the system.

Responses for implementation phase:



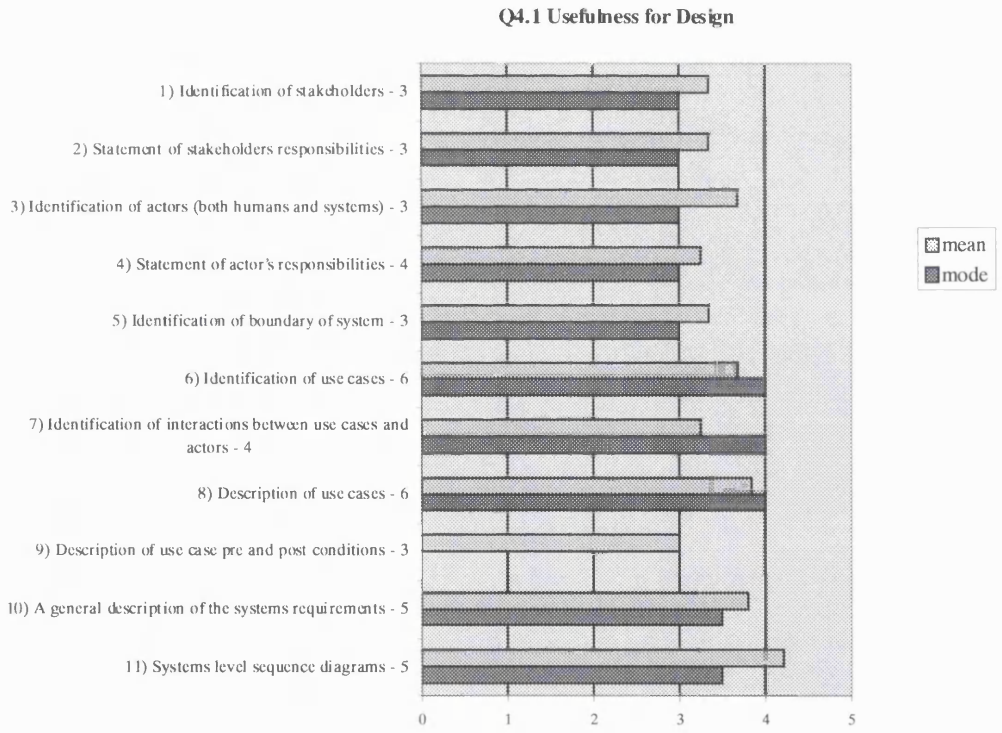
Responses for testing/integration phases:



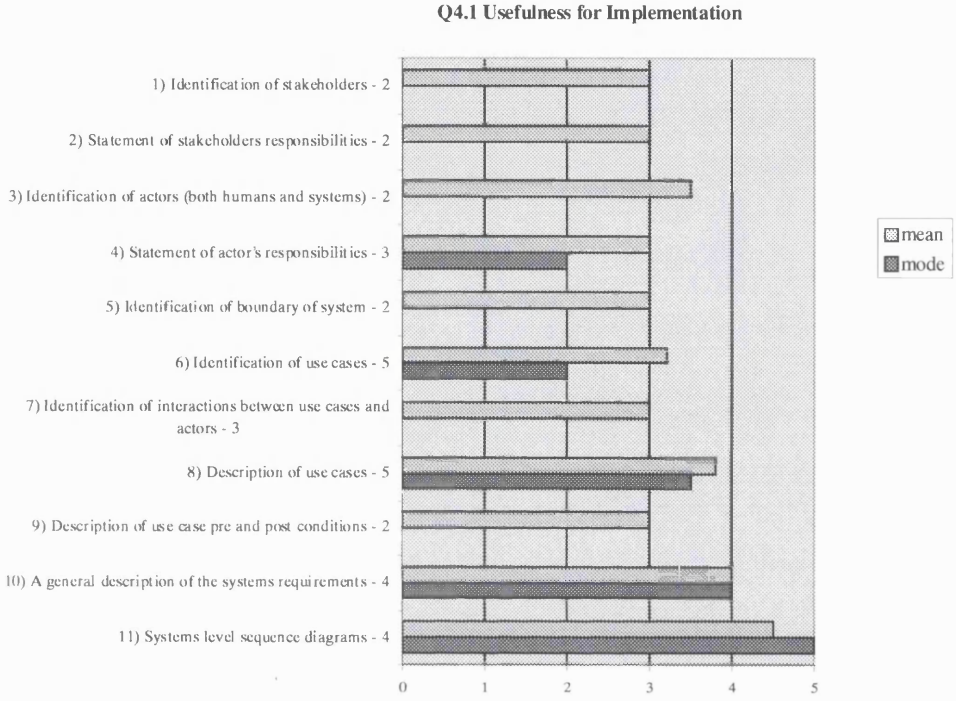
9.1.3 Responses from Sub-System Developers

Q4.1) Please state **how useful** you found the different parts of the **analysis specifications for components, systems or other sub-system** that interacted with your sub-system. Please indicate the usefulness to you of each part of the analysis specifications of other components and (sub)systems for both the design and implementation of your sub-system.

Responses for design phase:

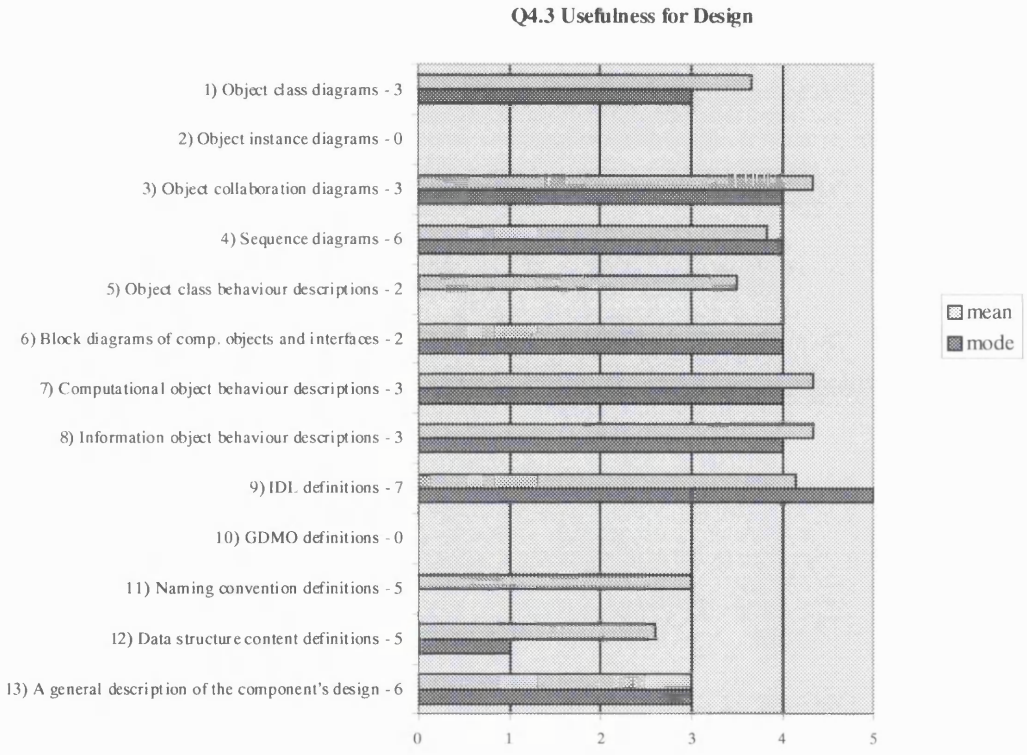


Responses for implementation phase:

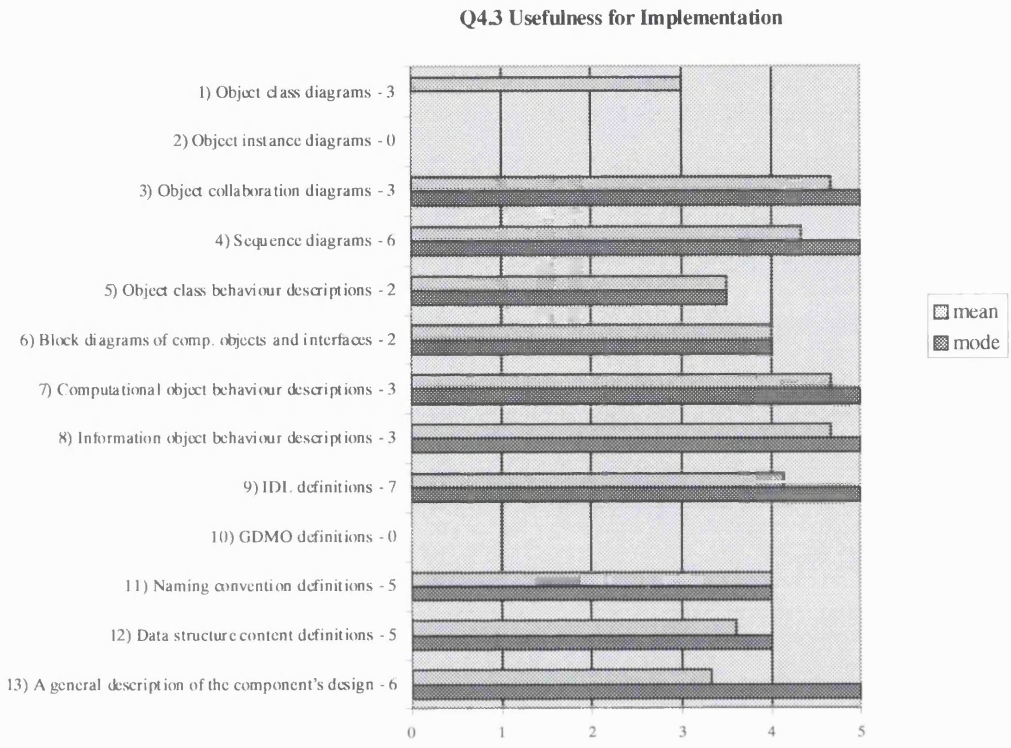


Q4.3) Please state **how useful** you found the **design specifications for components, systems or other sub-systems** with which your sub-system had to inter-operate. Please indicate the usefulness to you of each part of the design specifications for the design, implementation and testing/integration of your sub-system.

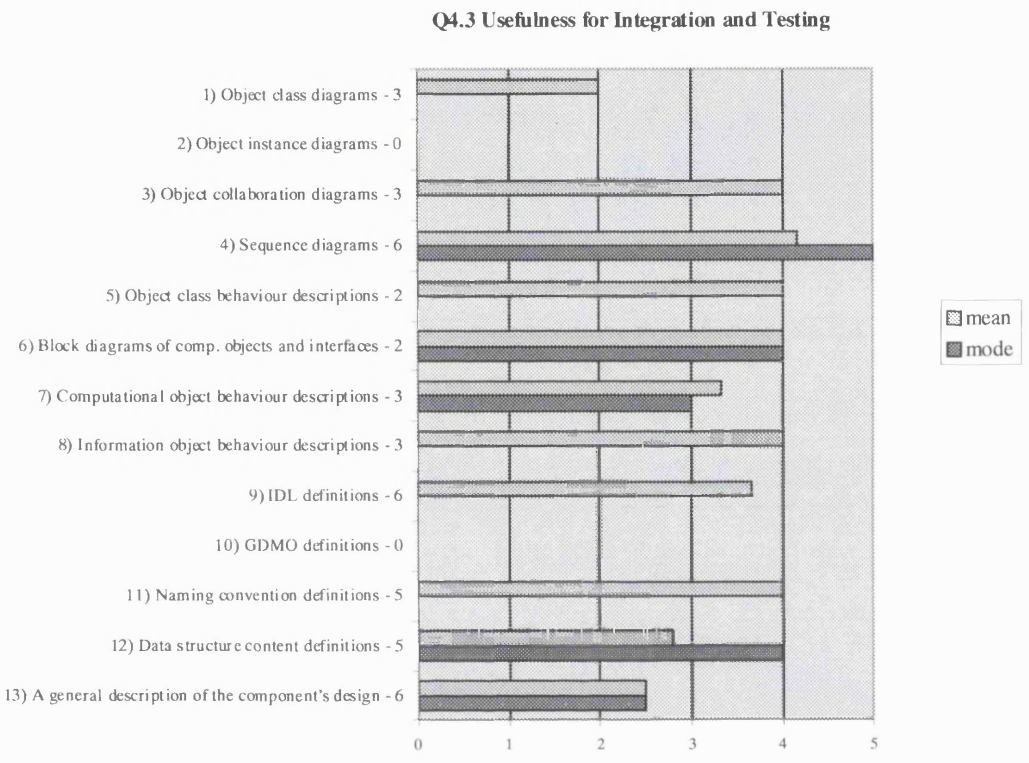
Responses for design phase:



Responses for implementation phase:

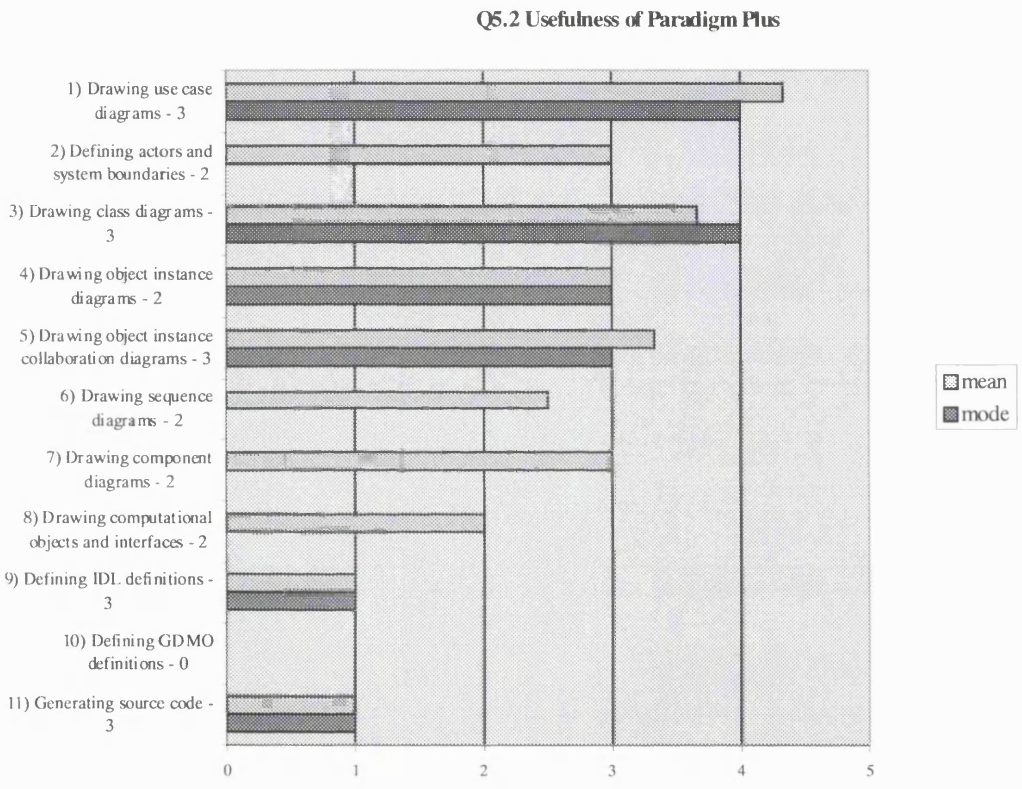


Responses for testing/integration phases:

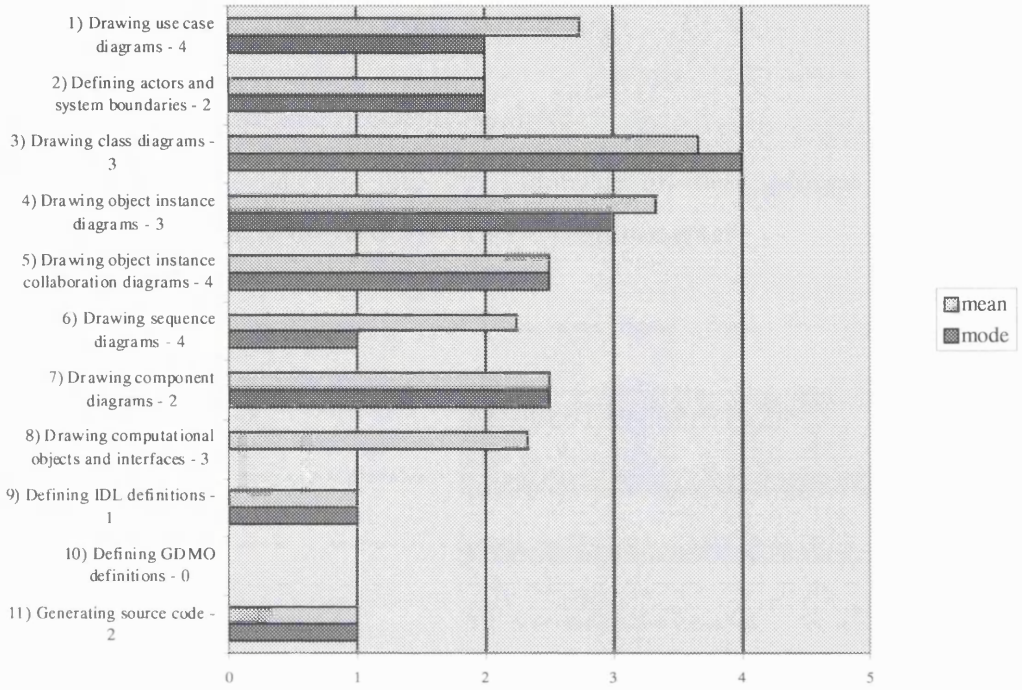


Responses on Tools Use

Q5.2) If you answered yes to any of the CASE tools in Q5.1, please give your opinion of how useful the CASE tool you used was for the following activities.



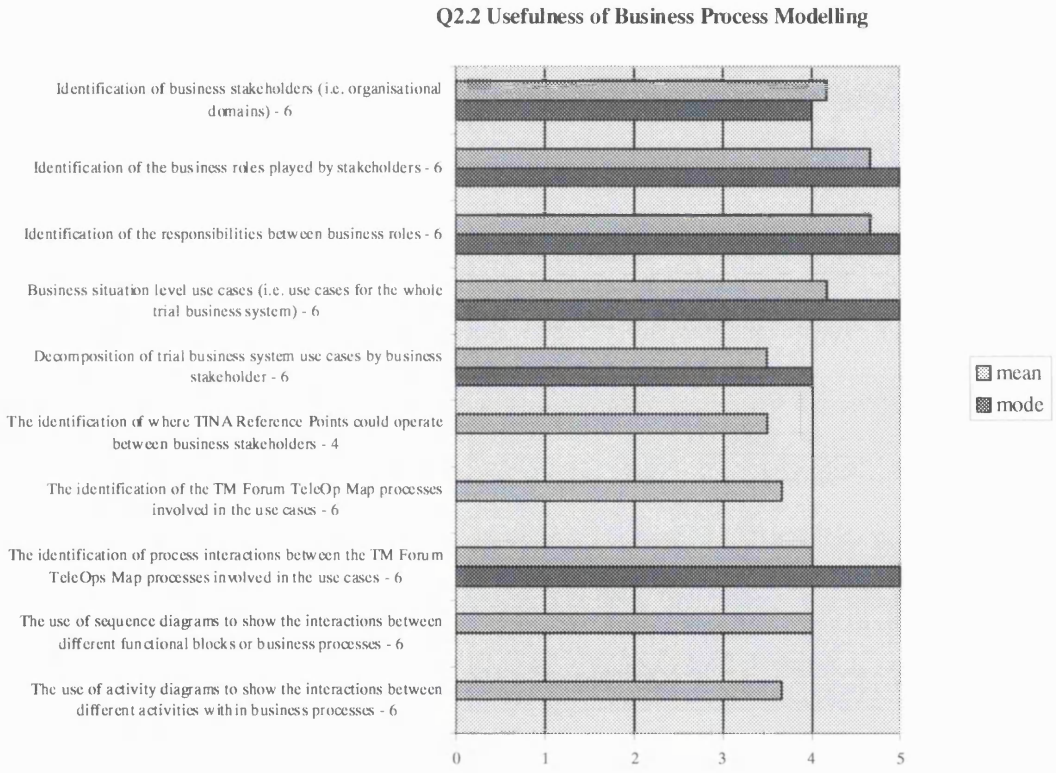
Q5.2 Usefulness of Rational Rose



9.2 Case Study 5 Response Summary

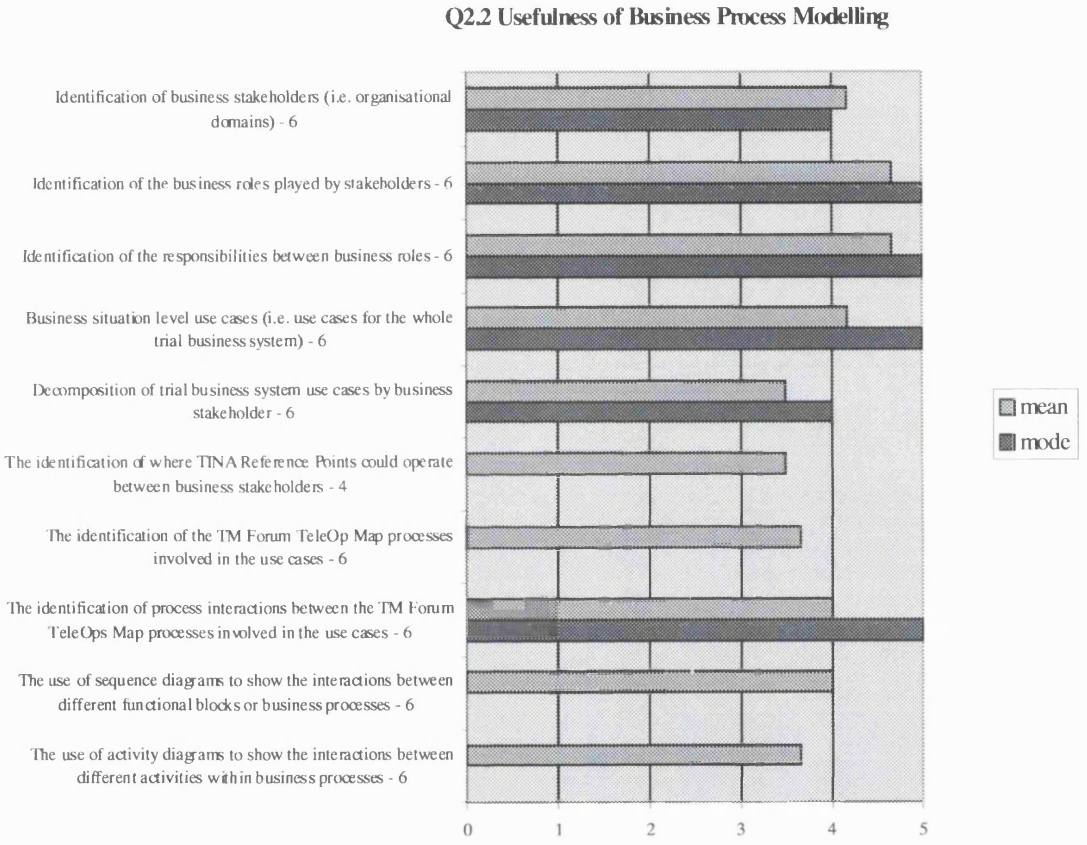
9.2.1 Responses from Requirements Analysts

Q2.2) How useful did you find the following **business process modelling constructs** in **analysing the business process requirements**?

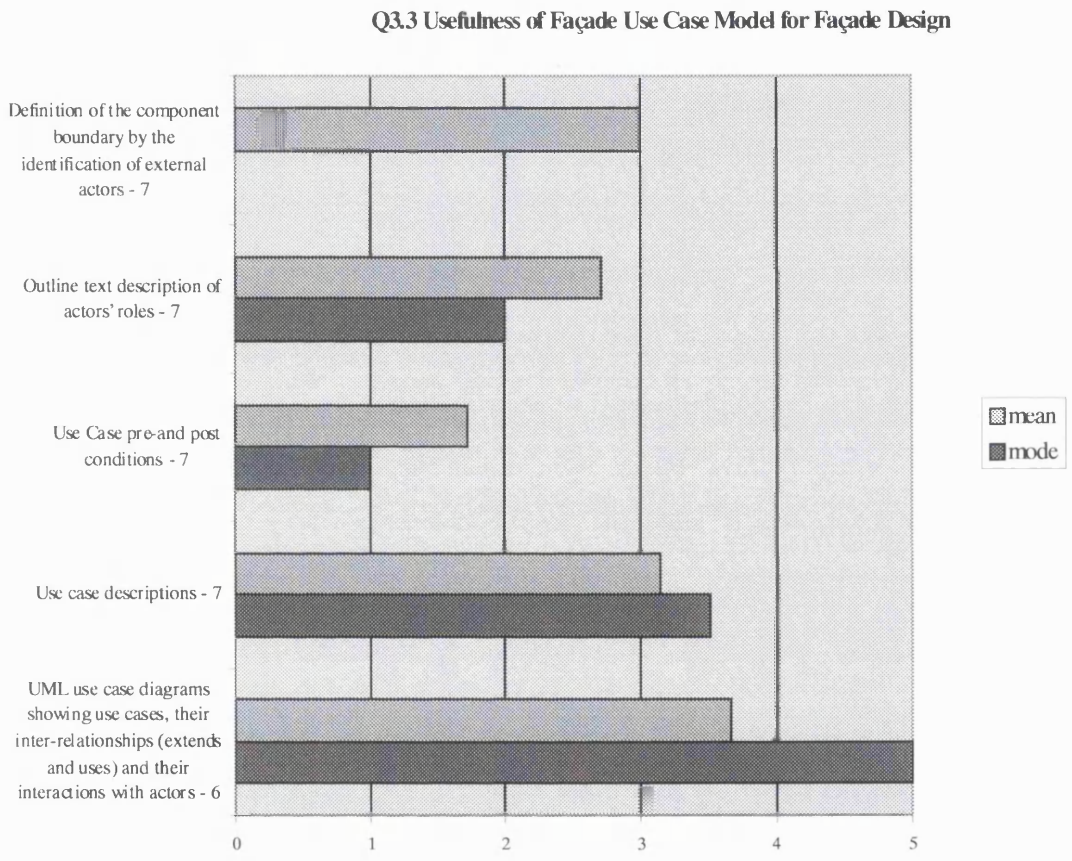


9.2.2 Responses from Component Developers

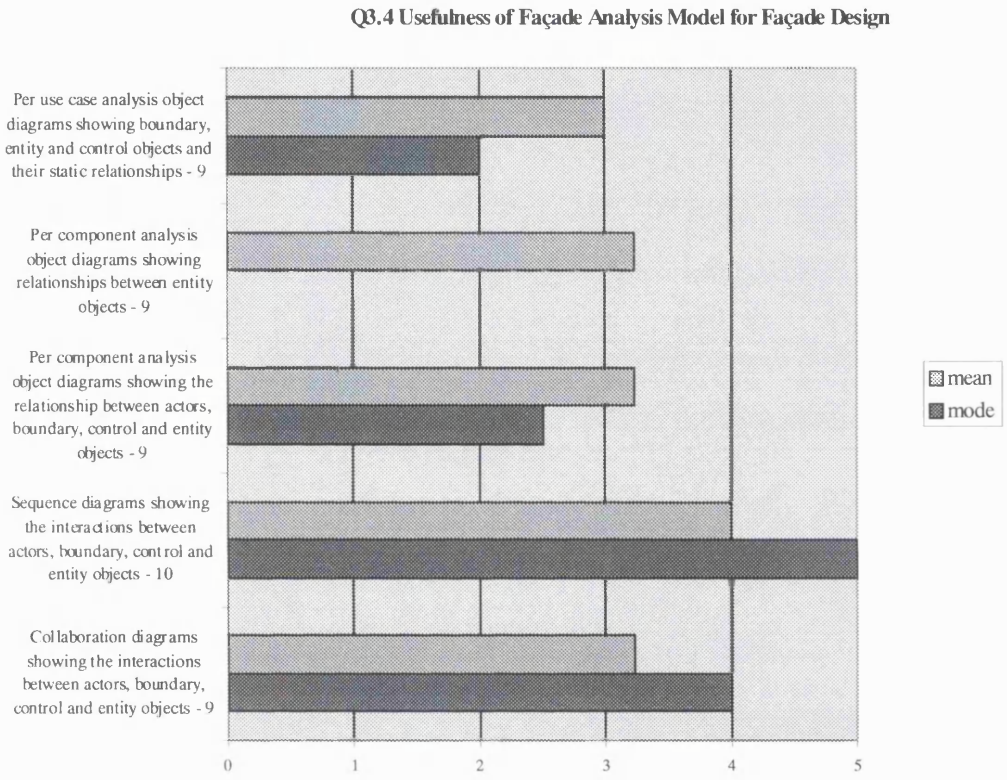
Q3.2) How useful did you find the following facade **use case modelling constructs** when documenting the facade **analysis model** for your component?



Q3.3) How useful did you find the following facade **use case modelling constructs** when documenting the facade **design model** for your component?

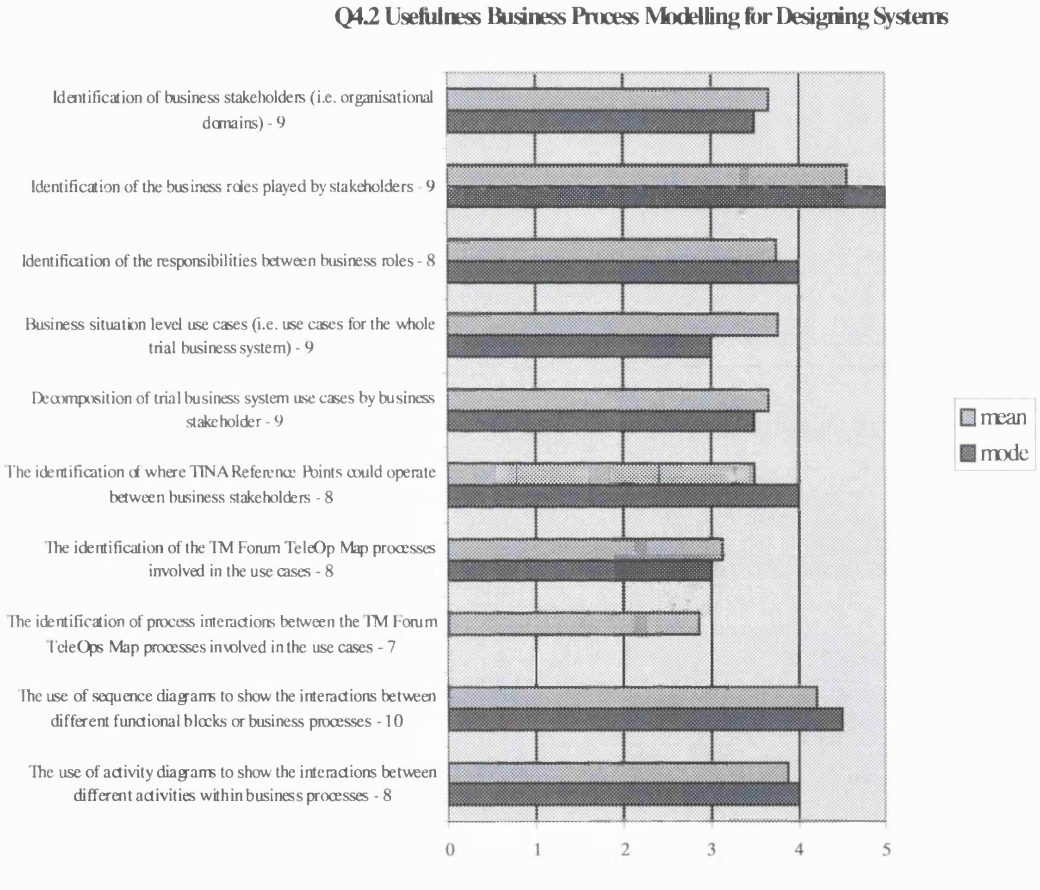


Q3.4) How useful did you find the following facade **analysis modelling constructs** when documenting the facade **design model** for your component?

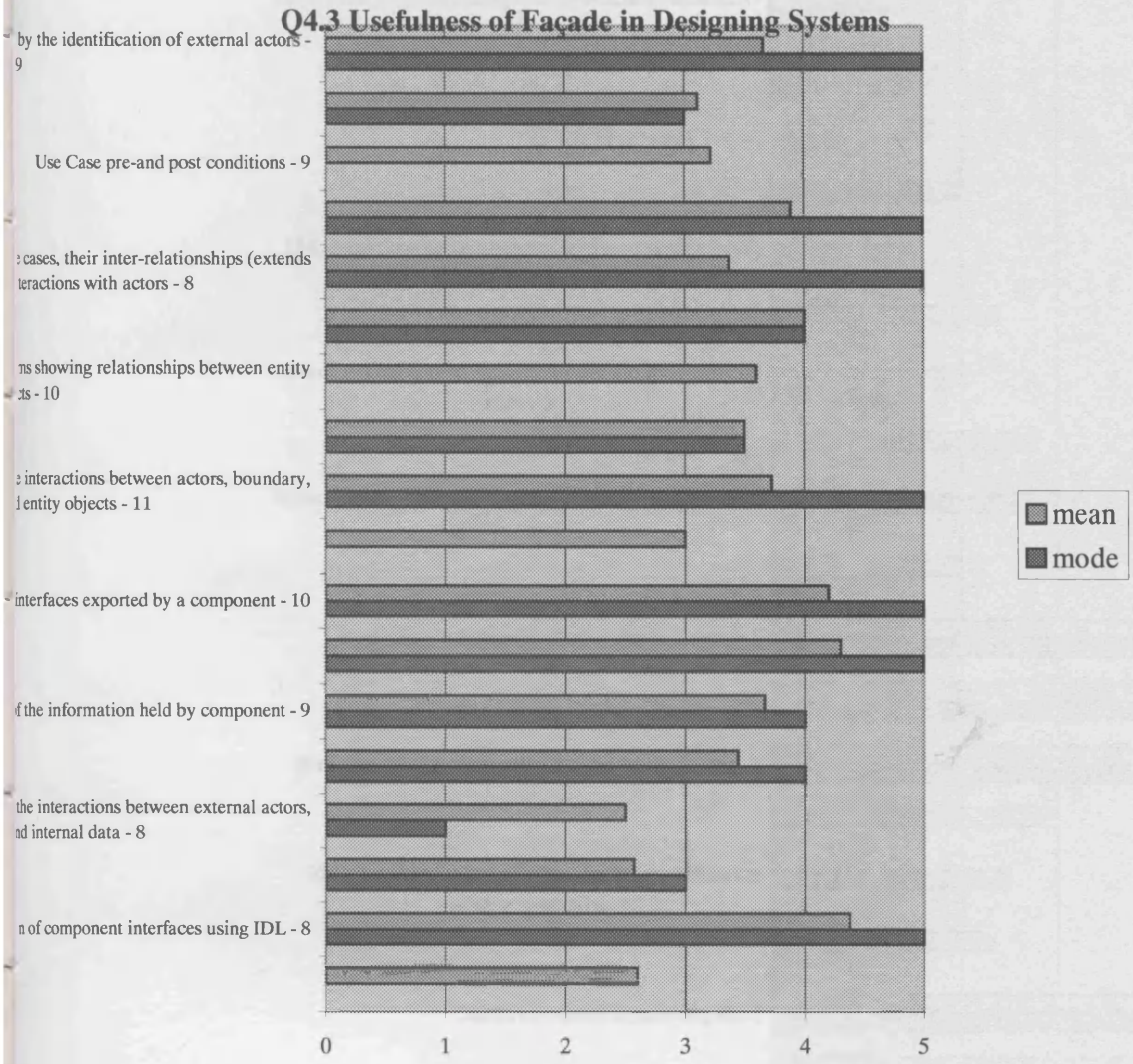


9.2.3 Responses from Trial Business System Developers

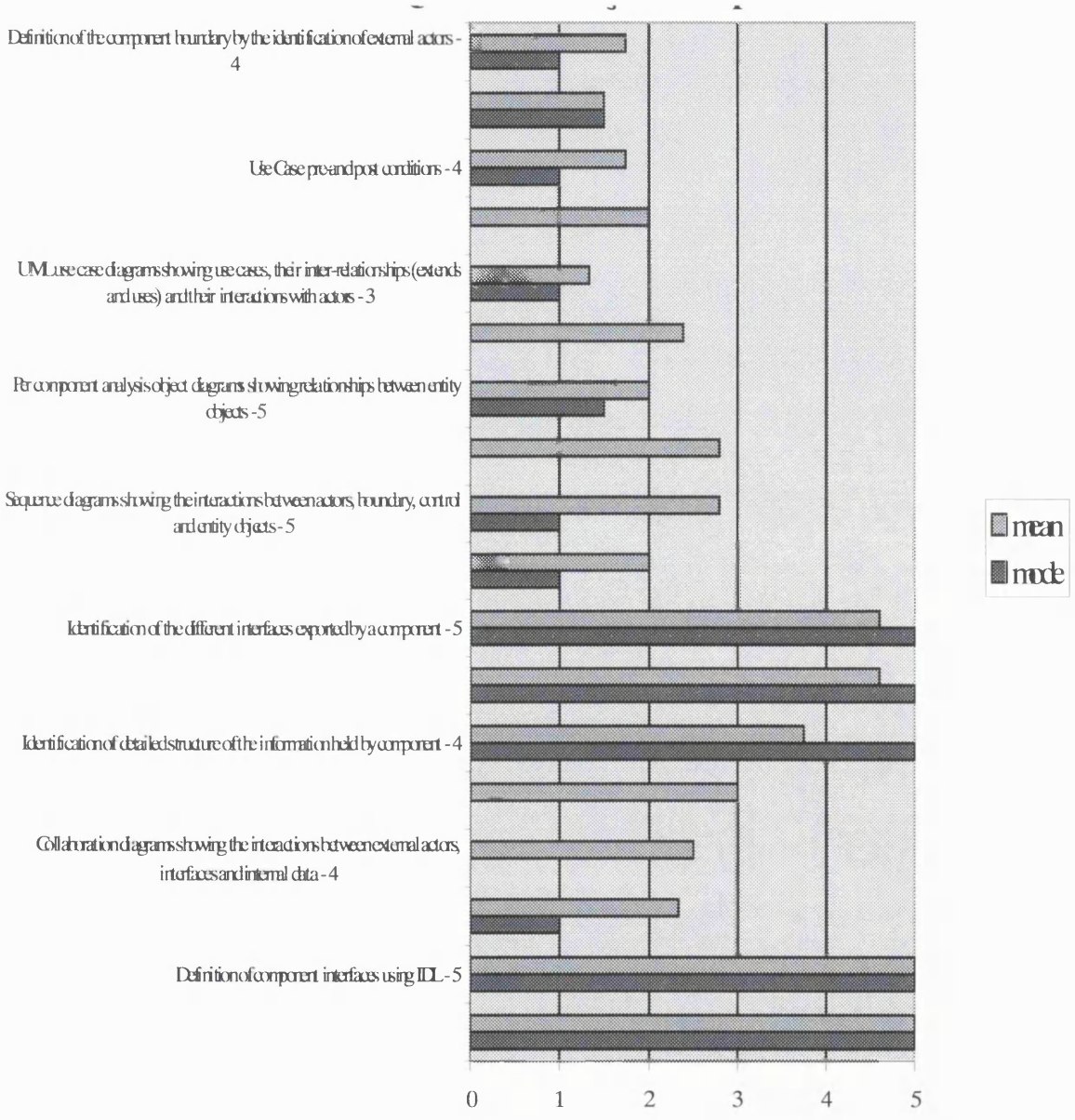
Q4.2) How useful did you find the following **business process modelling constructs** when **designing** the trial business system model?



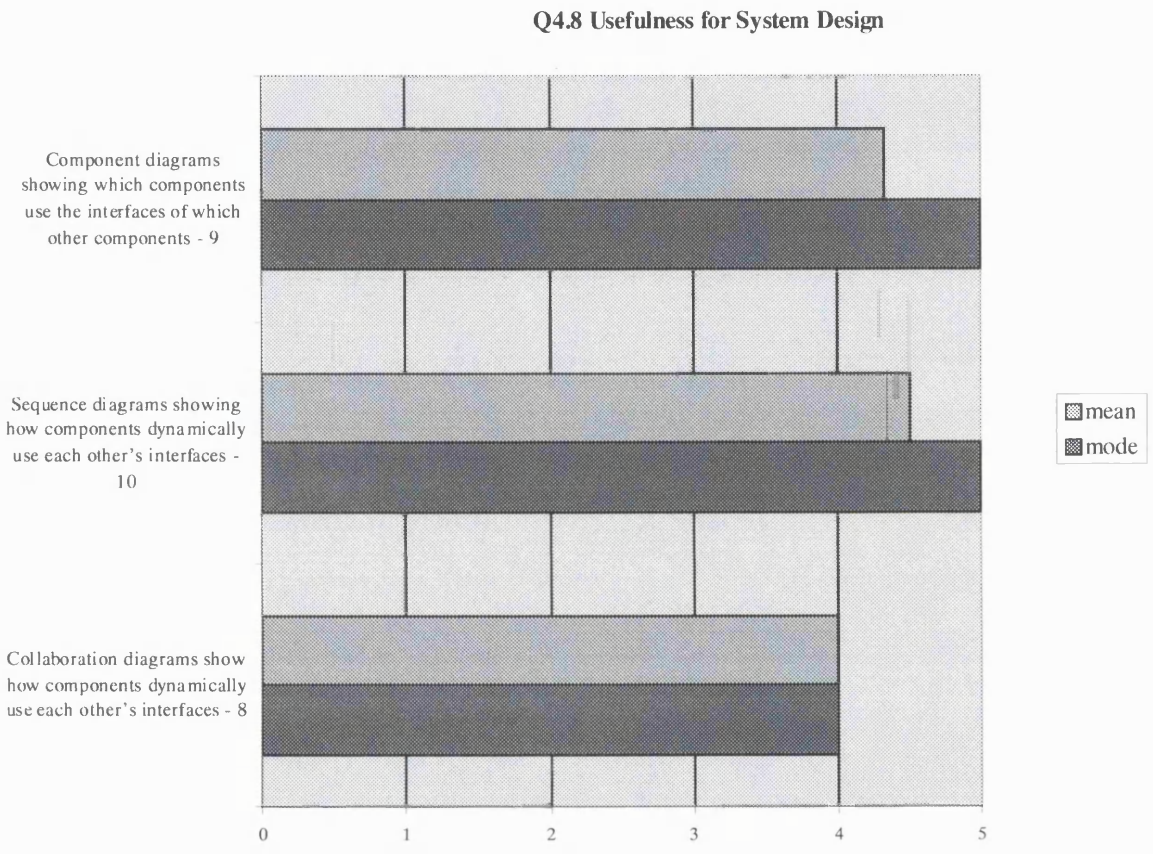
Q4.3) How useful did you find the following **facade model constructs** when **designing** the trial business system model?



Q4.6) How useful did you find the following **facade model constructs** when **modifying components** for reuse in the trial business system?

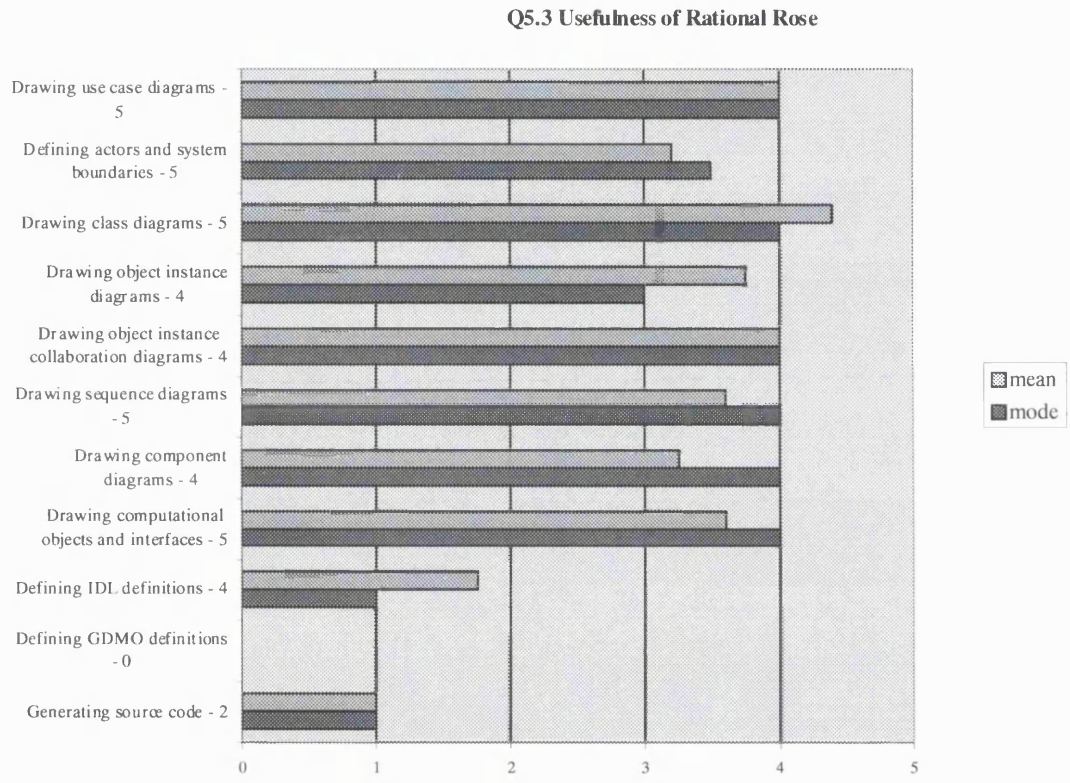


Q4.8) How useful did you find the following system modelling contructs when designing your Trial business System



9.2.4 Responses on Tools

Q5.2) Please give your opinion of how useful the CASE tool you used was for the following activities.



Q5.2 Usefulness of using Paradigm Plus