A DEVELOPMENT FRAMEWORK FOR OPEN SERVICE MANAGEMENT SYSTEMS

David Lewis
Department of Computer Science, University College London
United Kingdom

ABSTRACT

Management systems for communication services operate in a competitive, multi-domain environment in which open interfaces are essential for service providers in order to interoperate with customers and other service providers. Open interfaces and framework agreements are also required to take advantage commercial off the shelf components and so reduce the costs and time taken in management system development. This paper defines the requirements for a framework that will specifically support the development of open service management systems. Requirements are defined for a system architecture, a business model and a development methodology.

The system architecture consists of a functional framework, a technology framework and a component framework. An analysis of the current state of the art in these areas, as applied to service management, suggests that these frameworks should not be heavily prescriptive in order to accommodate the current heterogeneity in the market. Equally, with the telecommunications sector undergoing major structural changes, a well-defined set of business roles is seen as being of only limited value to management system developers. Therefore, it is suggested that business modelling for service management systems should be based on the modelling of business processes, as currently espoused by the TeleManagement Forum.

The development methodology, however, is suggested as an area where a common prescriptive approach can be of great benefit to open management system developers. Adoption of UML as a modelling notation will ease the communications between different stakeholders in the development of open management systems and the components they reuse. The proposed methodology emphasises the iterative nature of the development process and the pre-eminent place of component definition within it. Representing components at an analysis level as well as at a design level enables component reuse to become much more central to the analysis process and to become directly relevant to the business process reengineering activity.

KEYWORDS:

SERVICE MANAGEMENT, COMPONENT REUSE, BUSINESS PROCESS RE-ENGINEERING

1. INTRODUCTION

The world-wide deregulation of the telecommunication market has resulted in a rapid increase in the number and the dynamism of service providers and of the services they offer. However, the proprietary nature of many of the existing systems used to manage communications services is proving an obstacle to providers' ability to react flexibly and quickly to this rapidly changing environment. Service management systems are required to help reduce operating costs and to interact efficiently with customers and suppliers [1]. Service management systems must be able to quickly meet requirements as services evolve, but with tightly controlled development costs. Increasingly developers are turning to open system solutions in order to meet these challenges, principally in the following forms:

- *Open Management Platforms:* Service management systems are necessarily distributed because the service and network components they manage are distributed. Adopting open distributed computing platforms based on standards such as CMIP, SNMP, CORBA or HTTP greatly eases the task of integrating different provider's and vendor's systems.
- *Open Management Interfaces:* By agreeing standards for exchanging management information for specific management functions, the risks of tying commercial relationships to technical interfaces are reduced and the development of off-the-shelf solutions encouraged.
- *Open Components:* There is increasing interest in using industry agreements, both on open platforms and on open interfaces for specific problems, to support the development of a more open market in software components. It is envisaged that this will produce a large population of commercial off-the-shelf components that can be easily integrated into solutions to specific problems, greatly reducing bespoke development costs.

Developments in these areas are proceeding apace and across a wide range of different bodies, many of them general in nature and not specifically addressing the needs of telecommunications management. Developers of service management systems, therefore, will benefit from a framework that presents current and emerging open solutions in a manner that supports their problem domain directly. If such a framework is to provide comprehensive guidance to the developers of open service management systems it must address the following:

- *System Architecture:* to provide guidance on how to construct software systems from components originating from different sources, thus ensuring that these components are themselves structured so as to be easily integratable. A system architecture may also provide guidance on how different technologies should be integrated and how the problems of distribution should be addressed.
- *Business Model:* to provide guidance on which types of organisational stakeholders are typical of the service management problem domain, what forms of business relationships may exist between them and the functional scope of those relationships. Care must be taken that this is not too constrictive in modelling the structure of the fast changing telecommunications sector.
- *Development Methodology:* this is a set of processes and notations that enable the stakeholders in the development of open management systems to communicate their needs and capabilities to each other more clearly. It should also provides these stakeholders with a more rigorous, measurable development process aimed at improving the efficiency of their overall software engineering activities.

Many frameworks have been proposed for various problem domains within the telecommunication sector. The framework proposed here contrasts with some proposed in the area of network signalling and Intelligent Networks (IN) [2][3], which aim to provide a high level of automated support in the development of network control. Such service creation techniques rely on a high level of technological homogeneity and a stable, well-defined functional base from which to create new services. This does not reflect the situation in service management where a wide range of technologies are applicable, where business requirements are still not well understood and where there is no stable base to build on, e.g. no common model covering the range of services to be managed.

Instead the framework proposed here only loosely ties together the different technologies, architectures and business models involved. It

aims to provide an accessible knowledge-base to be used when attempting to construct open service management systems according to a tailored development methodology, rather than being the basis for a service creation environment.

The next section describes in more detail the stakeholders in the development of open management systems, and identifies the bodies of work existing in this area. Section 3 describes in detail how far existing standards go in satisfying the requirements for a System Architecture. Section 4 details an approach to Business Modelling, based on existing NMF and TINA solutions. Section 5 describes a suitable Development Methodology, building on existing approaches, but focussing on business process reengineering and component reuse problems in service management.

2. BACKGROUND

A model of the overall business context in which open service management systems are developed can help in our understanding of the requirements for a suitable development framework. The business model for the development of open service management systems centres on a management system developer stakeholder operating in a market where it provides management systems (possibly internally) to a service provider stakeholder. A management system must support one or more management tasks required by the service provider, which may involve interactions between the service provider and customer stakeholders and/or other service providers. Ideally, the development of management systems should make use of commercial off-the-shelf components, purchased from component vendor stakeholders in an open market. The system developer also relies on the use of open standards for platforms and common management functions. These ensure both the interoperability between the provider's systems and those operated by customers and/or other providers and the interoperability between components purchased from different vendors. The general management system development situation is summarised in figure 1.

Several bodies have already addressed problems in the area of service management. The distinction between service management and network management was recognised initially in the Telecommunications Management Network (TMN) standards. The TMN architecture [4] defines conceptual layers addressing different concerns within a provider's operation support structure. These layers are; a network element management layer, a network management layer, a service management layer and a business management layer. The TMN functional architecture makes distinctions between; network element functions; mediation functions; adapter functions to non-TMN compliant network element managers; workstation functions presenting information to human operators; and general operations system functions. It also makes distinctions between different reference points that may exist between these different types of functional units and between functional units within and outside of the same organisational domain. These reference points provide the basis for defining interfaces between implementations of the functional units. Initially it was assumed that these interfaces would be implemented using OSI Management, i.e. CMIP used to access Managed Objects (MOs) defined in GDMO. However, current revisions to the TMN standards are encompassing other technologies such as the CORBA from the Open Management Group (OMG) where MOs would be defined using the accompanying Interface Definition Language (IDL). The TMN family of standards also includes methodological guidance on the development of management interfaces [5]. This is based on the definition of management functions that are hierarchically decomposed into MO definitions. Management functions have been defined not only for general functions such as event management and log control, but also for network-oriented management functions, e.g. [6]. Though some of these standards can be reused at the service management layer, the ITU-T or OSI communities have defined few management functions specifically for this layer.

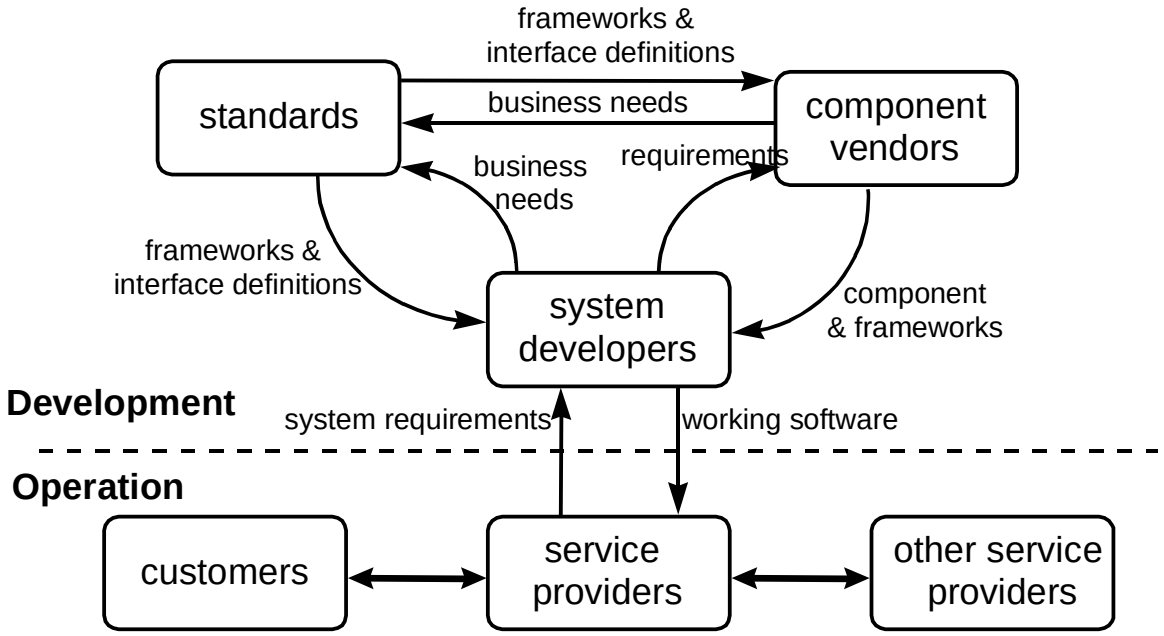

Figure 1: Business Model for the Development of Open Management Systems

One body that has analysed service management requirements more directly is the TeleManagement Forum (TM Forum), formerly the NMF. This industrial forum aims to build on existing TMN standards with business agreements and procurement guidelines that directly reflects the industry's short term needs. It has developed a business process model that, based on surveys of major service providers, provides a more detailed breakdown of the set of business processes that typically encompass a service provider's operations management activities. Interactions between processes in different providers are also identified. This model is intended for use as the basis for identifying the requirements for specific agreements on common interfaces and information models, the development of which is an ongoing activity within the TM Forum. The TM Forum has also been very active in analysing the application of different technologies to telecommunications management, as well as techniques for supporting migration and co-existence of different technologies, e.g. CORBA-CMIP gateways [7]. The TM Forum has produced its own internal guidelines for workgroups developing agreements on management interfaces [8]. This is similar to a general system development approach, and draws on analysis techniques such as use cases and graphical modelling techniques such as OMT class and sequence diagrams [9].

Another body that has performed in-depth studies of service management is the Telecommunication Information Network Architecture Consortium (TINA-C) [10]. This group has aimed to develop a comprehensive architecture for telecommunications control and management based on Open Distributed Processing (ODP) principles as defined by ITU-T in [11]. It has generated detailed models for the integrated control and management of multimedia services and broadband networks based on existing concepts from TMN, IN and ATM. Notable results in relation to open service management systems have been the development of a general business model and reference point scheme [12], the detailed modelling of specific service management functions [13], and the application of reusable object-oriented components in the architecture [14]. TINA-C has developed internal guidelines for modelling its systems. These are based on ODP modelling concepts, principally the use of the five ODP viewpoints that separate, enterprise, computational, informational, engineering and technology concerns. This has been supplemented with OMT class diagrams, sequence diagrams and simple block diagrams showing computational component structures and their interfaces.

Further work on development methodologies has been performed in recent European research projects. The EURESCOM project P.610 has performed case studies developing multimedia service management systems [15]. As with other projects, these case studies have made use of the Unified Modelling Language (UML) [16], a third generation object-oriented graphical modelling notation, combining concepts from many of its predecessors, e.g. OMT, and now subject to standardisation by the OMG. The case studies provided examples of the application of UML use case diagrams for capturing the requirements of management systems, and of UML class, sequence and component diagrams to the design of these systems. The ACTS project TRUMPET performed a case study of an inter-domain service management problem that used ODP viewpoints modelled using UML [17]. They found UML mapped well to ODP viewpoints, with use cases used for the enterprise viewpoint, class diagrams for the information viewpoint, component and sequence diagrams for the computational viewpoint and deployment diagrams for the engineering viewpoint. However, some problems were identified with UML's ability to represent ODP computational objects.

A more detailed study into development methodologies for service management was carried out in the ACTS project Prospect. This project implemented a series of multi-domain management systems in phases over three years, with the aim of reusing and evolving components between phases. A development methodology was adopted that was use case driven, and that made use of class diagrams, sequence diagrams, collaboration diagrams and component diagrams [18]. The process was applied to the analysis of multi-domain management scenarios and to the complete development cycle, from analysis to testing, of both single providers' systems and of individual reusable components. The methodology aimed to support the iterative application of the development cycle to these systems and components, as was required by the phased nature of the project. ODP viewpoints were initially used in this process, however problems were encountered with the separation between the information and computational viewpoints. Though the viewpoints were seen as useful for documenting a completed system, the tools were not available to provide the strong traceable links between information and computational objects that are needed if the design is to be modified through multiple iterations. The separation between information and computational viewpoints was therefore diluted, and both systems and components were designed using class diagrams, component diagrams and sequence diagrams that mixed computational and information object types. This provided the designers with the flexibility they required to express the design models in the way that most closely represented the solutions to the various tasks required of the system or component. Prospect used UML to represent its use case diagrams, class diagrams, sequence diagrams, collaboration diagrams and component diagrams. It also adopted conventions for the structure of use case descriptions, for the naming of components and for IDL specifications. The use of these notations by the different groups concerned with performing either multi-domain analyses, system development or component development, made communication between the groups much more straight forward. A questionnaire of these developers revealed that the use cases in particular enabled the different groups to understand each other's output more clearly.

The following sections describe in more detail how these existing approaches can satisfy the requirements for a development framework for open service management system, and also where more study is required.

3. SYSTEM ARCHITECTURE

Several, apparently competing, system architectures are available to the developer of open management systems. No attempt to synthesise them into a single system architecture is attempted here, instead their relevant features and commonalities are presented. The main features of a system architecture are analysed below in terms of a functional framework, a technological framework and a component framework.

3.1 Functional Framework

A functional framework is required to provide a well understood context into which specific functional solutions can be placed and where their relevance to the rest of the architecture can be easily understood. They are important in supporting the development of open interface

standards, by providing an overarching framework for the output of the on-going standardisation process. As has already been discussed, the most well established architectural standard in this problem domain, TMN, has a functional architecture that restricts itself to specifying functional layers relevant to telecommunication management, as well as to conventions for identifying reference points between functional unit in these layers.

The TINA Consortium has divided its overall architecture into Service, Network, Management and Computing areas. This has helped structure its working groups internally, with specific solutions concentrated in the Service and Network architectures, each tightly integrated with Management and Computing architecture concepts that were provided more as guidelines. A business model and a set of reference point between business roles has been defined to provide a mechanism to map business requirements to specific interface solutions, as explained in section 4.2. The TINA functional framework can therefore be classified as quite restrictive, being optimised to provide tight integration between its functional areas at the possible expense of broader applicability.

With CORBA, the OMG has taken a much broader and more loosely coupled approach to its functional architecture, reflecting a wider constituency, i.e. information technology in general rather than just its application to telecommunications. The Object Management Architecture (OMA) defines an Object Request Broker as the mechanism for distributed object communication, and defines different classes of interface standards under which any ongoing standards work must operate. These classes are; CORBA Services, which are regarded as basic common services required by most application; Horizontal CORBA Facilities which are higher level service definitions support recurring application needs; and Vertical or Domain CORBA Facilities which address problems common to specific application domains, including telecommunications.

Though these approaches differ, they can all be said to show some form of functional layering, where higher level functions, offering the developer problem-specific functions at a higher level of abstraction, build on the services offered by more generic lower layers.

3.2 Technological Framework

Though many standardisation efforts have begun with the assumption that a certain underlying technology would be used, e.g. CMIP with TMN and CORBA with OMA, there is a growing realisation that standards must support interoperability between the results of these parallel efforts. This is partially due to the proliferation of industrially led, technology specific fora that are separate from the traditional working groups of the ITU-T and OSI, e.g. OMG, ATM Forum. It is also supported by the wide acceptance of key interoperability technology, principally TCP/IP. The most comprehensive analysis of the many, apparently competing, technologies currently available for telecommunications management is being conducted by the TM Forum in the development of its Technology Integration Map [19]. This is based on a survey of current service providers' technology approaches and a synthesis of how technologies can be most optimally integrated, and where additional technology inter-working is required. Essentially, the result follows a three-tier or model-view-controller paradigm. It is expected that CMIP and SNMP will persist as the mechanism for controlling and determining the state of the network. However, CORBA is expected to be deployed at the network, service and business layers, as it more closely matches the peer-to-peer model of systems at this level and supports well the integration of legacy operation systems. It is also expected that business information will be consolidated in distributed databases accessed through SQL services, rather than residing in separate operational units. Finally, it is expected that customer and operations staff will increasingly access information through WWW-based thin-clients possibly using Java applets. This has the advantage of decoupling user location from operations systems and reducing application maintenance costs. Many of the technology interworking standards are already in place, e.g. CORBA-CMIP and CORBA-SNMP gateways and Java-IDL mappings.

Consequently, the proposed technology framework cannot opt for a single solution, but must encompass and bridge between the wide range of applicable technologies.

3.3 Component Framework

Many system architectures have traditionally focussed on supporting the definition of open interfaces, whether defined as a protocol or as an application programming interface. There is an increasing realisation however, that if the benefits of component reuse are to be fully exploited, standardisation frameworks need to address the definition of components. This goes beyond simply defining open interfaces, but must also cover platform dependencies. For instance, OSI System Management Functions provide standard interfaces to access functions in a CMIP agent. It is clear that very powerful agent operations can be provided by combining different functions, yet how their implementations interact within the agent is regarded as a platform specific issue, and left unstandardised. Thus it is not currently possible to buy platform independent implementations of these functions that can plug into different agent platforms.

TINA addresses this issue by specifying components according to its Universal Service Component Model. This requires a component to specify the different service interfaces it offers, which management interfaces it offers, and which services of other components or of the platform it uses. An extension to IDL, called Object Definition Language (ODL), is employed to group this information, and accompanying IDL definitions, together in a component specification [20]. ODL and its accompanying graphical notation are now being considered by ITU-T study group 10 for standardisation.

The OMG has also been addressing the use of components through its Request For Proposals (RFP) on a CORBA Component Model [21]. This aims to provide a component model that supports objects with multiple interfaces and the referencing of objects by value, in addition to specifying well defined mechanisms for describing components, handling component events, serialising component state and controlling component lifecycle. The OMG RFP also aims for compatibility with the Java component model, JavaBeans [22]. JavaBeans were initially only defined for application components, providing well-defined hooks for customising components and for integration with application builders. This has now been extended to server side components, termed Enterprise JavaBeans [23]. Enterprise JavaBean are components that are intended to run in a container that manages the lifecycle of components and provides them with support for remote interactions, naming, transaction, persistency and multithreading, freeing the component developer of these concerns. A component can be modified through extension classes, or at run time through property tables. The aim is that container platforms can be constructed from existing technologies such as CORBA, COM, or Distributed DBMS. The impact of component reuse on the development process is considered in more detail in section 5.2.

4. BUSINESS MODEL

Any development of open management systems needs to be driven by business requirements. A business model can be used in a development framework to provide guidance on domain specific business needs and business relationships when developing open interfaces or open components. A business model can also act as an entry point for system developers, who can compare their own particular requirements with the framework's business model in order to ascertain which existing open solutions within the framework may be applicable. Many different business models have been proposed by telecommunications system frameworks in the past, identifying business roles and the possible relationships between them. These models often reflect the different biases and concerns of the bodies proposing them. For instance the business models proposed by TINA-C and the EURESCOM P.610 project are fairly similar, but still differ in some types of business roles, e.g. TINA-C proposes a Broker role, while P.610 suggests an Access Network Provider role. However great care must be taken is adopting any specific business model. The deregulation of the global telecommunications market is leading to very rapid structural changes through mergers, collaborations (sometime imposed by regulation) and a wide range of new market entrants. This can quickly make parts of any specific business model irrelevant or inaccurate, reducing its usefulness to developers.

A more long-lived approach may be that taken by the TM Forum. Instead of attempting to provide well-defined business roles, the TM Forum has constructed a model of the various business processes that go on within service providers in general. These processes collectively represent the different possible functions of a service provider, and so any potential business role could be described by constructing sets of these processes. The following sections describe in more detail the

TM Forum business model and then demonstrates how it can combined with the more prescriptive TINA business model in building multi-domain management business models that can be readily analysed to make best use of existing open interface solutions.

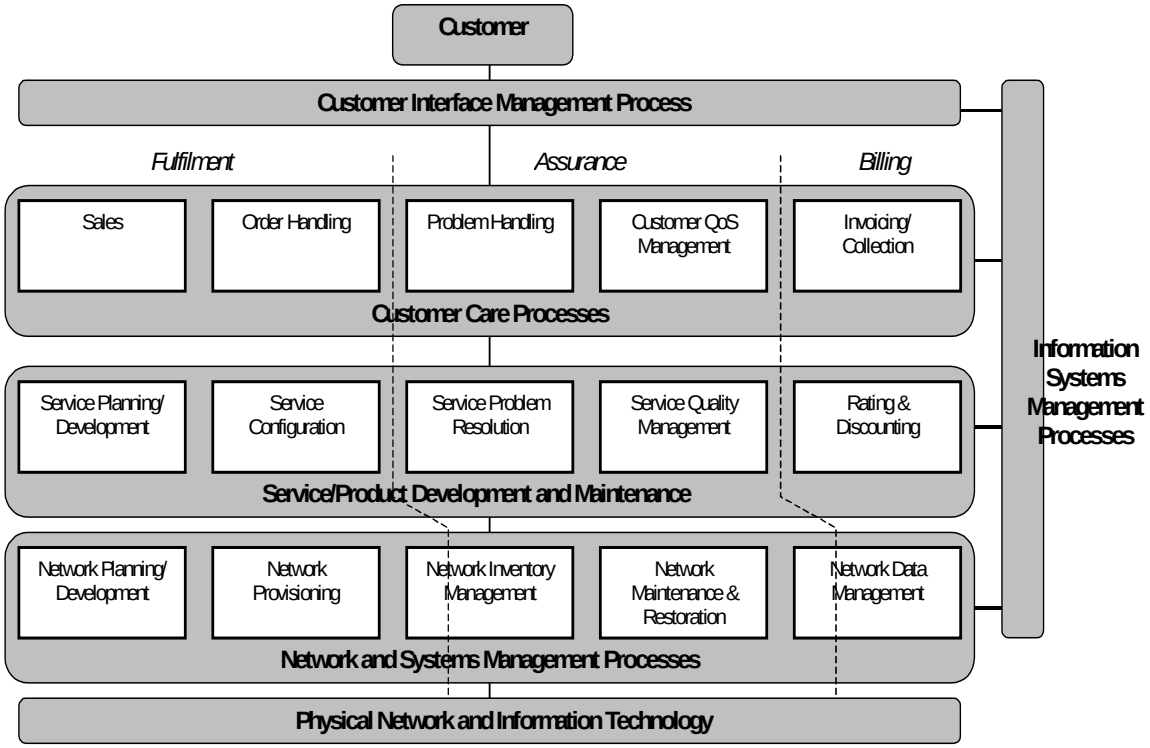

Figure 2: NMF Telecommunications Operations Map

4.1 TM Forum's Telecommunications Operations Map

The TM Forum business model is described in the TM Forum Telecommunication Operations Map [24]. The primary aim of this model is to provide a reference against which the definition of standardised interfaces between service providers and customers, suppliers or other service providers can be conducted. It was based on surveys of existing providers and framed so as to enable discussion of industrial agreements without having to expose the possibly sensitive internal process structure of any particular TM Forum member. A simplified view of the model is presented in figure 2. The model is partitioned horizontally into processes that relate directly to the customer, to internal service development and maintenance and to the management of the provider's networks and systems. The processes are also grouped vertically into major service management areas, i.e. the fulfilment/delivery of the service, the assurance/maintenance of the service and the billing/accounting for the service. Individual processes are defined in terms of activities within the process and of input and output triggers to the process.

4.2 TINA Business Model and Reference Points

The TINA approach to business modelling is to identify a number of business roles and define the reference points that exist between them. The TINA Business Model defines the following business roles.

- The Consumer role, which takes advantage of services provided by a TINA system but without the intention of providing TINA services to other business roles.
- The Broker role, which enables other business roles to locate service providers.
- The Retailer role, which is concerned with providing services to the Consumer role.

- The Third Party Provider role, which is concerned with providing services to Retailers or other Third Party Providers, but not directly to Consumers.
- The Connectivity Provider, which operates a network and provides connectivity services over it to other business roles.

A set of business relationships (see figure 3) is specified between these business roles. TINA reference points are defined in relation to the business relationships they support. All the non-Consumer roles have self referential business relationship supporting the federation of and business co-operation between these roles.

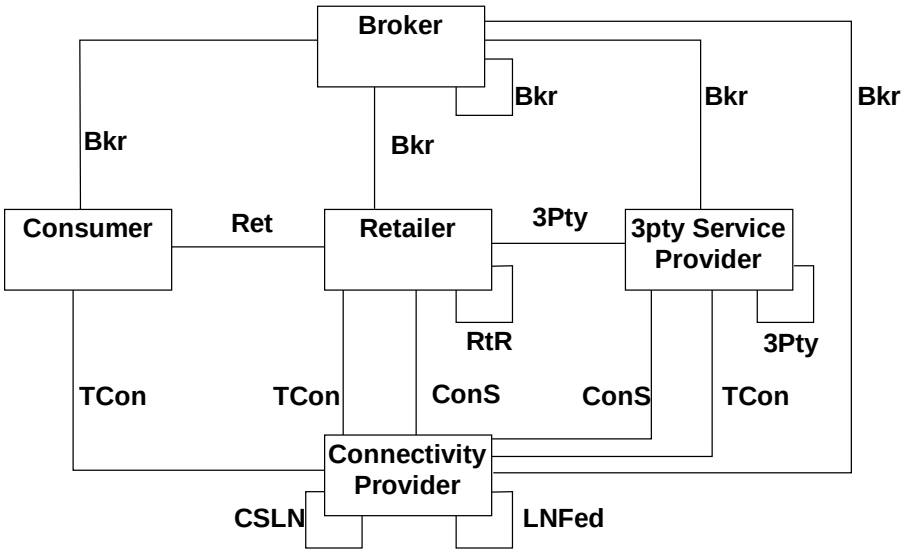

Figure 3: TINA Business Roles and Relationships

In a business situation where a TINA conformant system is required to support inter-domain interactions, the different business administrative domains involved may be characterised by the TINA business roles they play with respect to each other. This determines the TINA business relationships they have with respect to each other. The identification of business relationships then allows inter-domain conformance specifications to be defined by the amalgamation of the reference points related to these business relationships, plus any service specific interactions required. These reference points are defined in segments, with common segments used to cover the core parts of TINA system functionality. The primary segmentation is between access functionality and usage functionality. The access segment is concerned with the authentication and authorisation of users, the selection of services and the setting up the context for the use and management of services. The usage segment is subdivided into primary usage segments that cover the functionality that is the main objective of the service, and ancillary usage segments that address administrative and management functionality. This segmentation of reference point definitions enables any inter-domain reference point to be defined with the minimum set of functionality needed for the business relationships being analysed.

The TINA Service Architecture defines the details of interactions between components in terms of feature sets. These feature sets specify levels of functionality in a TINA system, such as basic session control or multiparty session control. Different feature sets are supported by interfaces provided by the individual software components defined in the TINA Service Architecture. These components are contained in a software architecture that supports their integration in different configurations to implement different groups of feature sets.

Feature sets can be mapped to reference point segments, so that the segments for any particular inter-domain reference point should determine, via the mapping to feature sets, the minimum set of

software components that would be required to conform to this reference point. The same principle can be applied between systems playing different business roles within a business administrative domain, i.e. at intra-domain reference points. This mechanism therefore provides a ready mapping from relationships in an abstract business model to the set of TINA software components needed to implement systems that provide the required TINA conformant interface functionality. In addition, TINA allows for the run-time negotiation of the feature sets to be used between two parties.

Currently, however, there is no publicly available mapping of initial TINA reference point segments to feature set definitions. When available, this will only cover the service independent session control and management functions specified in the Service Architecture. The intention is to expand this collection of mappings with further service-specific feature sets as they are defined by designers of specific TINA services. TINA's approach mapping business level relationships to component capabilities is considered further in section 5.

4.3 Mapping TM Forum Business Processes to TINA Business Roles

Though TINA defines a fairly restrictive architecture, it has been demonstrated that certain service control and management concepts can be applied to other frameworks, e.g. Internet services [25]. Mapping the more flexible TM Forum business model onto the TINA one may therefore enable us to determine where specific TINA management solutions can be more widely applied. Such a mapping also provides us with an example of how the TM Forum's business model can be used in the analysis of other management frameworks.

Before examining such a mapping however, the core differences between the two models must be appreciated. Firstly, the TM Forum's Operations Map defines general business processes in existing service providers. These may be human based processes or automated ones. Part of the intention of the Operations Map 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 TM Forum's Operations Map 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 Distributed Processing Environment (essentially CORBA) will be used to implement reference point interactions, while the TM Forum's Operations Map makes no assumptions about implementation technology (this is addresses in the TM Forum'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 TM Forum 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 influenced the initial TM Forum Operations Map to a large extent. Finally, the TM Forum's Operation Map 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 TM Forum's Operations Map to the TINA business model and reference points is to identify which TM Forum processes operate in which TINA Business Roles. Note that some TM Forum processes may be present in more than one TINA Business Role.
An initial mapping of the TM Forum business processes onto TINA business roles is given in the figure 4.

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 also 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 process 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.

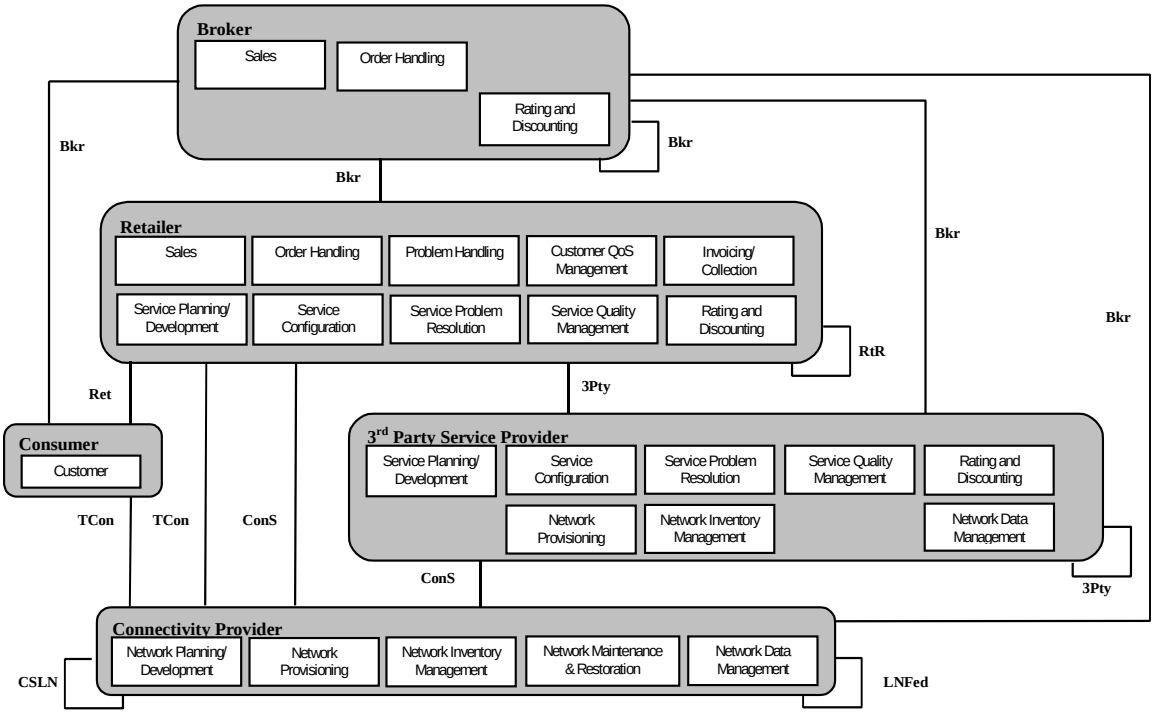

Figure 4: Mapping of NMF Business Processes onto TINA Business Roles

This mapping is currently being considered as a basis for future TM Forum/TINA-C liaison. The subsequent steps to reconciling the results from these two bodies depend on the relative maturity of the TM Forum inter-process links and the TINA business relationships. Full TM Forum business process specifications are given in TM Forum Solution Sets, while agreements on TINA business relationships are defined as reference point specifications. However, both these sets of standards are under development, so there is an opportunity to use the mapping between the two business models to determine where TM Forum solutions sets can influence TINA reference points specifications and vice versa. Given the possibility of merging the TM Forum and TINA models, one pressing issue is to align the notation and process used in defining both TM Forum solution sets and TINA reference points. Though both aim to define open interfaces between business roles, the TM Forum approach is based on analysing business process information flows, while the TINA approach involved integrating component interfaces into suitably modular segments. A suitable methodological approach that could be applied in concert with the integrated TINA/TM Forum business model is addressed in the next section.

5. DEVELOPMENT METHODOLOGY

The guidelines presented here for the framework's development methodology are based on the assumption that the development process consists of the following major activities:

- Requirements Capture: This is the activity where the requirements of the system customer and users are elicited and recorded as a set of use cases and non-functional requirements. The output of this activity is a Requirements Statement.
- Requirements Analysis: This is the activity where the Requirements Statement is analysed and models are generated that; breakdown the system requirements into different functional blocks, identify the informational requirements of the systems and identify the interfaces to the system. The output of this activity is the Analysis Model
- Design: This is the activity where the Analysis Model is made more detailed and modified to support non-functional considerations and other implementation related aspects. The output of this activity is the Design Model.
- Implementation: This is the activity where the Design Model is translated into software code.
- Testing: This is the activity where the implemented software is tested against its functional and non-functional requirements.

Each activity is capable of being the source of changes to the system, either to correct an error or an oversight in the output from a preceding activity, or due to factors that only become apparent at the level of detail encountered in the current activity. As a result all activities in the chain may have an impact on the output of preceding activities.

The activities tend to be started in the order given above, but they will necessarily overlap in time as later development activities impact on the activities that started earlier. Typically the output of each activity is mapped from and builds upon the model from the previous one. The management of change in the development process therefore requires traceability between the models. The more comprehensive the traceability between elements in one model and elements of the models from the subsequent activities, the easier it is to propagate the effect of changes across the models. For instance if objects in the design model are individually traced back to objects in the analysis model, then the impact of a change in the design model and can be readily reflected in the analysis model. This overall development process is depicted in a highly simplified form in figure 5. Obviously, the later in the development process an error detected or a change is made the wider the ramifications for the different models used in the development process, and therefore the more difficult and expensive the change becomes.

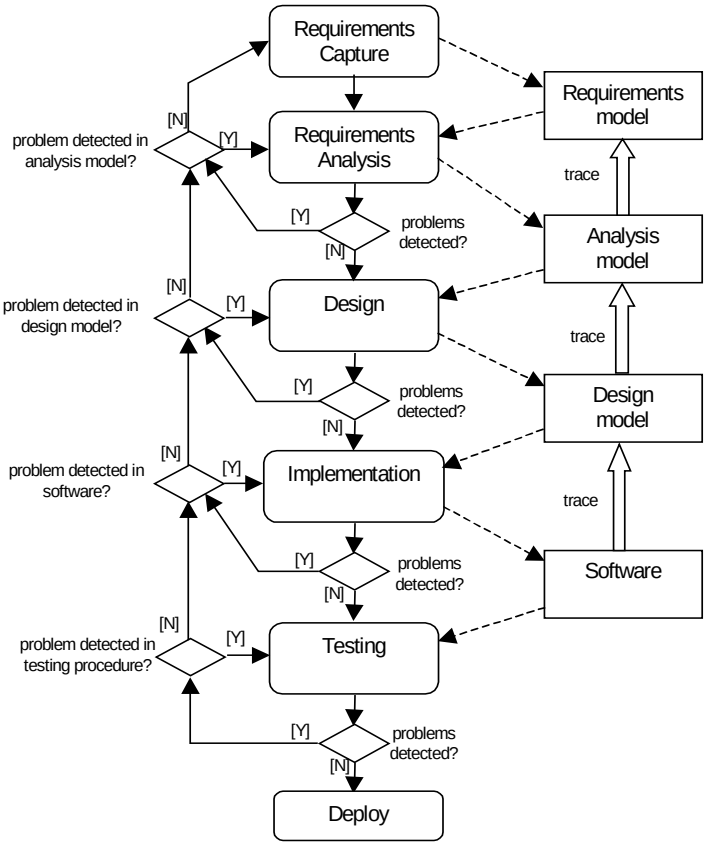

Figure 5: Model of Development Process Activities

Experience from the Prospect project suggests that the application of use cases is useful in developing management systems. Use cases provide a powerful mechanism for ensuring each stage of development is kept focussed on the overall systems requirements. Unlike telecommunication control systems, the design goals of management systems tend to focus on the delivery of management information to human operators, a strength of use cases, while the modelling of real-time and concurrent behaviour, a weakness of use cases, is not so relevant.

Jacobsen's Object Oriented Software Engineering (OOSE) methodology [26] provides the required mechanism for mapping requirements expressed in use cases to an analysis model. Use cases are text descriptions of an interaction the user has with a system from which he or she gains benefit. Analysis modelling involves analysing the user's requirements as represented by use cases, in order to define the high-level type structure of the system. The analysis model concentrates on trying to define the static types of the system, their groupings and their relationships to each other. The aim of the analysis model is to build robustness into the design of the system at an early stage, principally by careful modelling of the high-level structure in a way that minimises dependencies between different model components and maximises the opportunity for reuse. Analysis modelling does not concern itself with considerations such as data base management systems, distribution mechanisms, programming languages, existing products or performance. These are deferred to the design activity.

The analysis model refines the use case model by linking use cases to analysis objects. OOSE specifies three general class stereotypes for analysis objects, following the model-view-controller paradigm. These are designed to aid the robust structuring of the system. These stereotypes are:

- Entity Objects: These are long-lived objects that represent the information content of the system. Typically they may be involved in several use cases instances.

- Boundary objects: These handle the communication between the system and its surroundings.
- Control Objects: These perform use case specific behaviour that is not specific to boundary or entity objects.

The UML representation of these stereotype classes is suggested in [27] and recently proposed to the OMG as an UML extension in [28]. A slightly different representation available with the Paradigm Plus CASE tools is shown in figure 6.

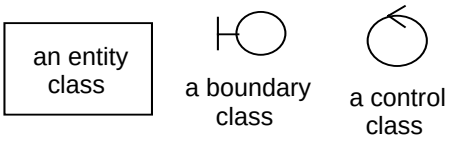

Figure 6: Analysis model stereotypes

In mapping use cases to an analysis model; entity objects will be typically derived from noun phrases that occurs 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. Figure 7 is an example of a use case diagrams showing the analysis objects related to a couple of use cases for a subscription management component developed in the Prospect project. Interaction diagrams are often required at this stage to detail the lifecycle of objects and dynamic aspects of the relationships between them.

The analysis model is refined into a design model. This 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. The experiences reviewed in section 2 show that UML class diagrams, sequence diagrams, collaboration diagrams and component diagrams all play an important role in developing design models.

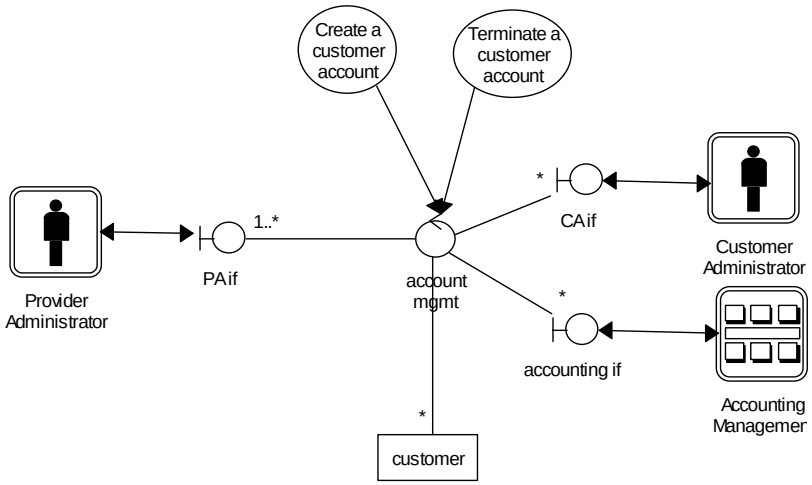

Figure 7: Use case diagrams showing analysis objects

5.1 Business Process Modelling

While OOSE provides good traceability through the development cycle of a system, at the analysis stage use cases only help us analyse the interactions between a system and the actors that use it at the system boundary. Use cases are not good at describing the internal operation of a system. Even if sub-systems are identified, and use cases for each subsystem generated, these use cases may only interact with external roles, or subsystems modelled as roles. There is no well-understood mechanism for tracing the interactions between different use cases in different subsystems. Use cases in different subsystems may be related by generalisation relationships, e.g. “uses” or “extends”, but these do not define details of the interactions, only that they are related. Use case text can include details of interactions with

different subsystems, but this quickly becomes unwieldy, generating in effect a textual description of the system's internal functionality.

The identification of interactions between subsystems however, is typically the kind of analysis that is performed in business process reengineering activities. Here, businesses aim to define the major processes they provide to their customers, which at a high level can be adequately captured with use cases. However, the aim of business process reengineering is to analyse the internal processes of an organisation to understand how they interact to provide value to customers, and how the structure of these processes and their interactions can be changed to improve customer services and reduce costs. This problem is complicated for service management systems by the interactions often required with processes in other organisations. The framework's development methodology must therefore address the problem of analysing the requirements for a management task that must be performed by interactions between management processes in different domains, some of which will support automated interactions and some of which will not.

The TM Forum's Operations Map provides us with a model of suitable business processes which we are fairly confident reflects the typical operations of a service provider. This can therefore provide us with a starting point for the analysis and design of common solutions to management problems, e.g. the solution sets produced by the TM Forum. It may also, however, help us to analyse a service provider's business processes in order to identify where existing solutions, available as reusable components can be applied. 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. This may involve different events being triggered under different conditions and thus different sequences of activities being followed in the execution of a task. Such an analysis may represent parallel activities, the conditions for their completion and the synchronisation of control. Specific activities may also be broken down hierarchically into finer grained activities.

A common representation of such control flow is event-driven process chains, a graphical modelling technique that allows activities to be associated with organisational roles and with objects representing business information. As described in [29] the inclusion of activity diagrams allows UML to support a similar type of modelling diagram. 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. The use case may not necessarily mention the internal business processes. These processes therefore need to be analysed using activity diagrams. The activities can be placed within swim-lanes representing TM Forum business processes, possibly residing in different administrative domains. This will ease the identification of which existing TM Forum business agreements match the requirements of the task at hand. An example of a UML activity diagram taken from the FlowThru project is shown in figure 8. It represents the activities involved in subscribing a new customer to an ATM service.

5.2 Modelling for Component Reuse

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 above development process the component is therefore presented in terms of its design model and the software. This may cause problems in the development process, since changes required to accommodated the reuse of components are only likely to become apparent during the design process, therefore possibly countering aspects of the analysis model.

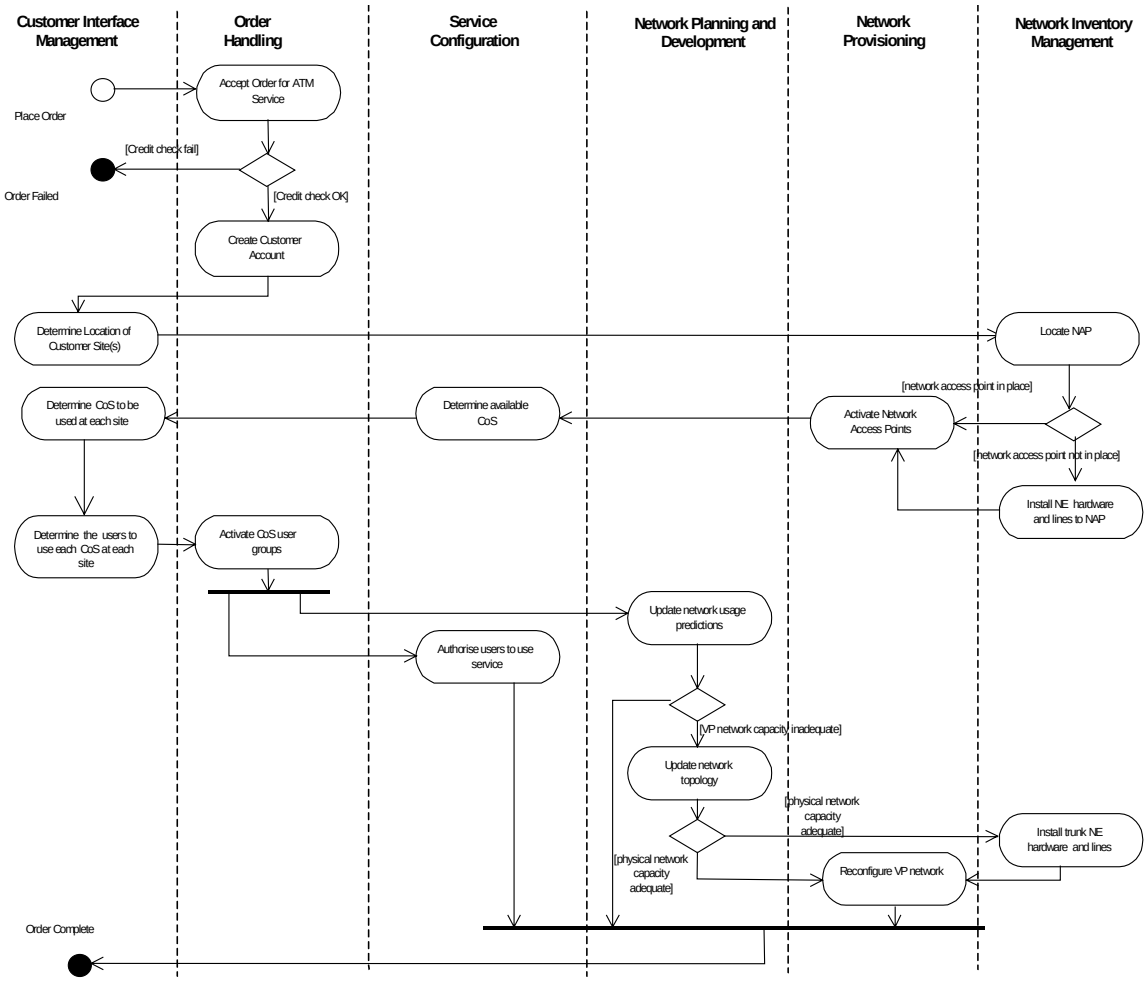

Figure 8: Example of UML activity diagram showing subscription to an ATM service

Often the component may be 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 therefore considered at the analysis stage in order that the system's analysis model is structured in a way that will accommodate the inclusion of the framework's components in the design stage. This situation is depicted in figure 9a. 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 telecommunication management systems development, such a typical component reuse situation is difficult to standardise because, as described in section 3, 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, it attempts 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 also at making decision based on architectural and functional aspects of a component rather than its 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 [27]. The basis of the approach is that components are not presented just as units of design and of software. Instead, they should be packaged together with the analysis model of the component, rather than being strongly integrated into a specific component framework. If the modelling techniques used for the analysis model of the component are similar to those used for modelling the system in which it might be included, then it becomes much easier for the analyst to assess whether the component is suitable for use in the system. In addition the system's analysis model can directly use 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 9b.

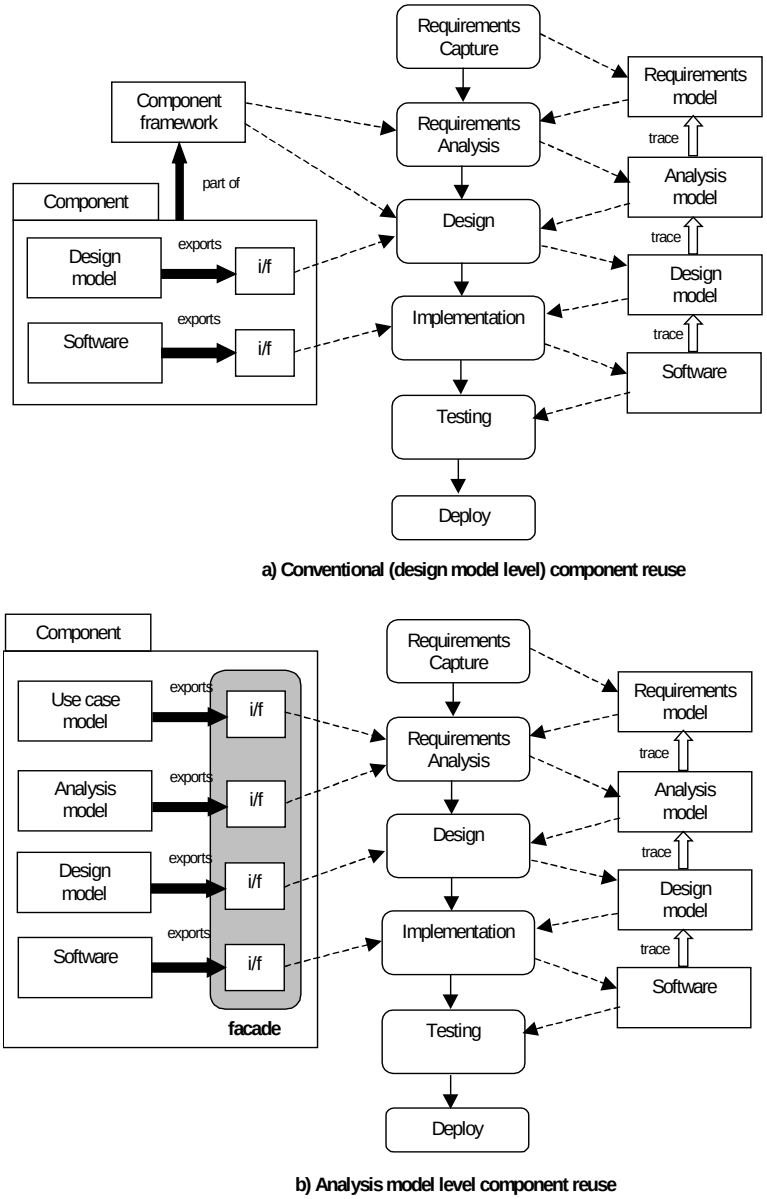

Figure 9: Different approaches to component reuse

The presentation of a component for reuse is known as a facade. A facade presents the re-user of a component only with 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. In the analysis model based reuse approach the facade consists not just of reusable code and the design model, but also the analysis model relevant to the design model part of the facade.

A component may present several different facades, possibly aimed at different types of re-users, e.g. black-box or white-box re-users. 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.

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 this development. Strong traceability between the component's internal models will ease the selection of parts of the model for inclusion in the facade based, for instance, on a set of use cases. However, it is not a requirement for the component to have been originally developed and documented using a OOSE-like process, and 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 the 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.

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.

If components are designed to satisfy the requirements of complex management information flows, they will necessarily have interactions with several external actors, possibly other components, preferable through open interfaces. Where this is the case, a use case description involving several external actors may not be sufficient to help analyse the inclusion of the component into a wider, system level information flow. In such a situation, the facade may be supplemented with activity diagrams that represent more clearly the component's behaviour when interacting with multiple external actors. These actors would be modelled as external activities, possibly mapped to activities in the framework's TM Forum based business model. This would make the component's role in a wider information flow more apparent. The input and output event triggers and associated information in a façade would therefore enable the component to be more easily integrated in a business process re-engineering activity.

The flexibility of a component is also key to it reusability. This can be captured using a variety of variability mechanisms. Typical variability mechanisms that can be used in a façade are class inheritance, use case generalisation, extensions to existing use cases or objects (at both analysis and design models) and parameterisation using templates, frames or macros. An obvious avenue of further research is to examine how the JavaBeans and CORBA Components model

variability mechanisms, and how these can be clearly represented in UML facades.

6. CONCLUSIONS

The need to develop telecommunications management systems rapidly and at increasingly lower cost provides a compelling argument for the wider use of open systems, where openness is applied to platforms, inter-domain interfaces and components. To develop and fully exploit such open systems, a domain-specific development framework is proposed. This framework is comprised of a system architecture, a business model and a development methodology for the development and reuse of open service management systems.

The range of standards and technologies that are applicable to open service management system development point to the adoption of a fairly loose system architecture. This will provide minimal functional structuring, while supporting technological heterogeneity and the emergence of a market in reusable components. Such an approach reflects the direction of the OMG's Object Management Architecture, the TM Forum's Technology Integration Map and the proposed component architectures for CORBA and JavaBeans. It is expected that revisions to TMN may also follow this approach.

Similarly, the fast changing structure of the telecommunication market precludes a long-lived business model based on pre-defined business roles. Instead a business model is adopted that is based on the definition of business processes and their interactions as proposed in the TM Forums Telecommunications Operations Map. This approach can be flexilby mapped to different business role models and demonstrated by the mapping given to the TINA business model.

A suitable development methodology is one area where a common approach can be of great benefit to open management system developers. Adoption of UML will ease the communications between different stakeholders in the development of open management systems and the components they reuse. The proposed methodology emphasises the iterative nature of the development process and the pre-eminent place of component definition within it. Representing components at an analysis level as well as at a design level enables component reuse to become much more central to the analysis process and to become directly relevant to the business process reengineering activity.

Though this development framework is based on existing standardisation efforts and application experience from recent research projects, further work is required to fully validate these concepts. The ACTS project FlowThru is applying the development methodology guidelines directly by integrating several existing management components into a number of different management scenarios.

ACKNOWLEDGEMENTS

The work presented in the paper was conducted with partial funding of the European Commission under the FlowThru project (AC335). The views expressed here are not necessarily those of the FlowThru consortium.

REFERENCES

[1] Adams, E.; Willetts, K.; The Lean Communications Provider: Surviving the Shakeout though Service Management Excellence; McGraw-Hill; 1996.

[2] TOSCA Deliverable 6: Service Creation: the TOSCA Paradigm and Framework Approach-AC237/BT/DS/P/019/B1; 1997.

[3] Efremidis, S.; Prevedourou, D.; Demounem, L.; Milsted, K.; Zuidweg, H.; TINA-oriented Service Engineering Support to Composition and Federation, Proceeding of 5th International conference on Intelligence in Service and Networks; Antwerp; Belgium; Springer-Verlag; 1998.

[4] Principles for a Telecommunications management network; ITU-T Recommendation M.3010; 1996.

[5] TMN Interface Specification Methodology; ITU_T Draft Revised Recommendation M.3020; 1994.

[6] Generic Network Information Model; ITU-T Recommendation M.3100; 1992.

[7] Inter-Domain Management Specifications: Specification Translation; X/Open Preliminary Specification; X/Open; Reading; Draft of April 17, 1995.
[8] Vincent, A.; Hall, C.; Modelling/Design Methodology and Template; NMF Internal Document; Draft 4; 18th October 1997.
[9] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.; Object-Oriented Modelling and Design; Prentice-Hall; Englewood Cliffs; N.J.; 1991.
[10] Chapman, M.; Montesi, S.; Overall Concepts and Principles of TINA; TINA Baseline Document TB_MDC.018_1.0_94; December 1994.
[11] Information Technology- Open Distributed Processing- Reference Model- part 1: Overview; ITU-T Draft Recommendation X.901/ ISO/IEC Draft International Standard 10746-1; 1995.
[12] Reference Points and Business Model; TINA-C; Version 3; June 1996.
[13] Service Architecture; ed. Kristiansen, L.; TINA-C; version 5.0; 1997.
[14] Natarajan, N.; Dupuy, F.; Singer, N.; Christensen, H.; Computational Modelling Concepts; TINA Baseline Document; TB_A2.HC.012_1.2_94; February 1994.
[15] Nesbitt, F.; Counihan, T.; Hickie, J.; The EURESCOM P.610 Project: Providing a Framework, Architecture and Methodology for Multimedia Service Management; Proceeding of 5th International conference on Intelligence in Service and Networks; Antwerp; Belgium; Springer-Verlag; 1998.
[16] Booch, G.; Rumbaugh, J.; Jacobson, I.; UML Documentation Set 1.1; Rational Rose; 1997.
[17] Kande, M.; Mazaher, S.; Prnjat, O.; Sacks, L.; Wittig, M.; Applying UML to Design an Inter-Domain Service Management Application; Proceeding of UML'98; Mulhouse; France; June 1998.
[18] Wade, V.; Lewis, D.; Donnelly, W.; Ranc, D.; Karatzas, N.; A Design Process for the Development of Multi Domain Service Management Systems; Guidelines for ATM deployment and interoperability; S. Rao (editor); pages 88-103; Baltzer Science Publishers; 1998.
[19] NMF Technology Map; Draft NMF GB909; July 1998.
[20] Graubmann, P.; Mercouroff, N.; Engineering Modelling Concepts (DPE Architecture); TINA Baseline Document; TB_NS.005_2.0_94; December 1994.
[21] CORBA Components: Joint Initial Submission. OMG TC Document orbos/97-11-24; Draft; November 1997.
[22] Englander, R..; Developing Java Beans; O'Reilly; 1997.
[23] Thomas, A.; Enterprise JavaBeans: Server Component Model for Java; White paper; http://java.sun.com; December 1997.
[24] NMF Telecoms Operations Map; NMF GB910; Stable Draft 0.2b; April 1998.
[25] Lewis, D.; Tiropanis, T.; Integrating TINA into an Internet-based Services Market, Proceeding of 5th International conference on Intelligence in Service and Networks; Antwerp; Belgium; Springer-Verlag; 1998.
[26] Jacobsen, I.; Christerson M.; Jonsson P.; Overgaard G.; Objected-oriented Software Engineering: A Use Case Driven Approach; Addison-Wesley; 1992.
[27] Jacobson, I.; Griss, M.; Jonsson. P.; Software Reuse: Architecture, Process and Organisation for Business Success; ACM Press Addison Wesley Longman; 1997.
[28] UML Extension for Objectory Process for Software Engineering; version 1.1; OMG; September 1997.
[29] Allweyer, T.; Loos, P.; Process Orientation in UML through Integration of Event-Driven Process Chains; Proceedings of UML'98; Mulhouse; France; June 1998.