A Software Development Methodology for Service Management

David Lewis

dl@uhc.dk

UH Communication A/S, Denmark

Abstract: Software development frameworks typically consist of logical and technological architectural guidelines coupled with methodological guidelines. The lack of a common logical architecture for service management and the range of technological solutions applicable to this area retard the definition of an open development framework for service management. The effectiveness of any common development framework may therefore depend more on the common applicability of the its methodological guidelines than on that of its architectural guidelines. It is therefore proposed that a common methodological approach to service management system development will be more effective than attempting to synthesis a set of common architectural guidelines. Specific methodological guidelines have been validated though implementation projects in the EU-funded ACTS program and are presented here as UML modelling constructs.

Keywords: Service Management, UML, Software Development Methodology

1 INTRODUCTION

Service Management is an area that is currently not well addressed by standards. The TMN family of standards focus largely on network and network element problems, with few standards available that are applicable to service management. Correspondingly, there is little evidence that TMN interface development methodology [m3020] is applicable to service management problems. Two industrial bodies that have addressed service management development in some detail at the Telecoms Management Forum (TMF) and the Telecommunications Information Networking Architecture Consortium (TINA-C). Both these bodies, in addition to developing some specific service management interfaces, have also provided some guidance on how Service Management Systems (SMS) could be developed within their particular architectural frameworks. The TMF has suggested an approach to open service management interface development that draws heavily on current object-oriented analysis and design techniques and notations. In particular the TMF promotes the use of use cases and graphical modelling using the OMG's Unified Modelling Language (UML) [vincent]. This is done with a framework of telecoms management business processes (termed the Telecoms Operation Map or TOM) used to identify which management tasks should be analysed to develop industry interface agreements. The TINA-C development approach is heavily based on the use of the five viewpoints specified in the ITU-Ts Open Distributed Processing (ODP) recommendations [x901]. This makes heavy use of multi-interfaces distributed objects (computational objects), which, by separation into service-specific and service-independent sets, provides strong support for software reuse when implemented in CORBA.

Both of these approaches have merit, the TMF one for its focus on addressing the real need of industry through analysing business processes, and TINA-C's for its support for component reuse. This paper presents a methodology that attempts to combine the best of these approaches to developing SMS. This methodology integrates the analysis of business processes and the design and implementation of systems built largely from reusable components. Maximum leverage has been made of existing software engineering techniques and tools, with UML used throughout. Aspects of business process re-engineering techniques, and their application using UML activity diagrams, have

been adopted for matching system requirements to component capabilities. This matching task is supported by modelling components at high levels of abstraction using constructs based on the widely used Object Oriented Software Engineering (OOSE) technique proposed by Ivar Jacobsen [jacobsen97].

The techniques suggested for this methodology have been validated and refined though SMS development in several EU-funded ACTS projects [lewis95][lewis99c]. The specific techniques proposed in this paper have been trialed and validated in a current EU project, FlowThru. Here three telecommunications management scenarios have been analysed, design and implemented using the methodology. Each scenario represents process interactions from one the process areas identified in the TMF TOM [nmf-gb910], i.e., fulfilment, assurance or billing (the second of these is being demonstrated at the EU stand in the Telelcoms'99 Expo). The implementation of each scenario has been constructed from existing management components, reused integrated using the proposed modelling constructs.

2 BACKGROUND

An analysis of the problems facing SMS developers [lewis99a], suggested that this area can benefit from an approach to modelling and integrating SMS that is commonly understood by SMS developers, Commercial Off-The-Shelf (COTS) Software Vendors and developers of service management standards. These stakeholder types must interact in developing their products, i.e. either SMS, COTS software for SMS or interface standards, as indicated in Figure 1. Such interactions have therefore been identified as points of communication where a common development methodology would be most beneficial.

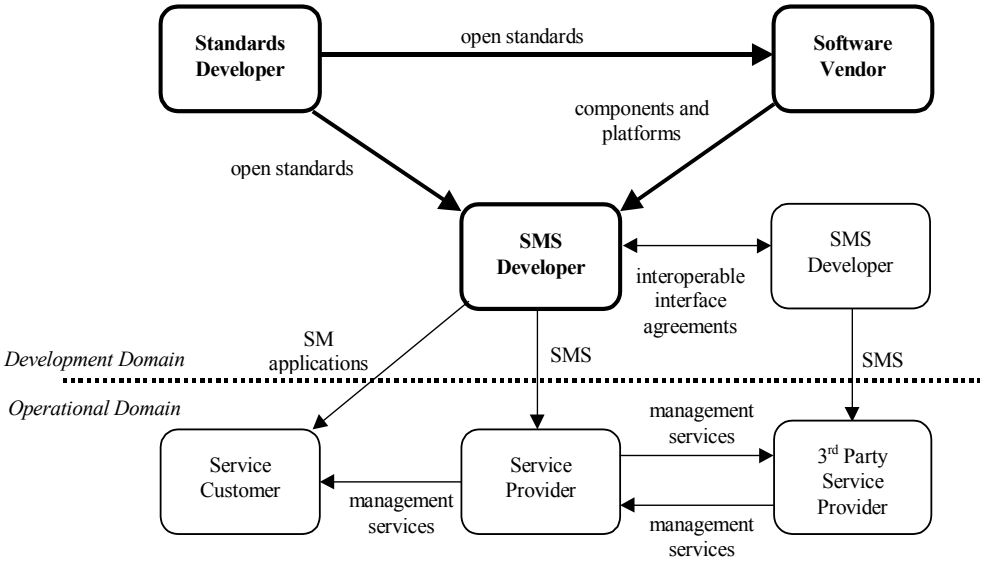
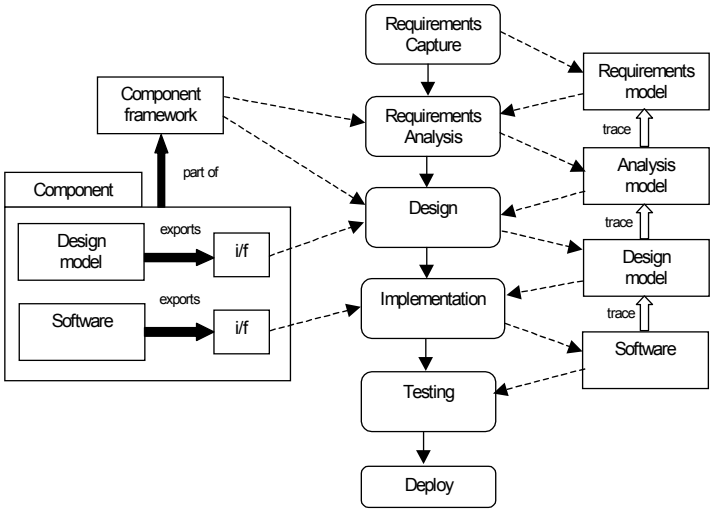


Figure 1: Relationships between stakeholders in SMS development

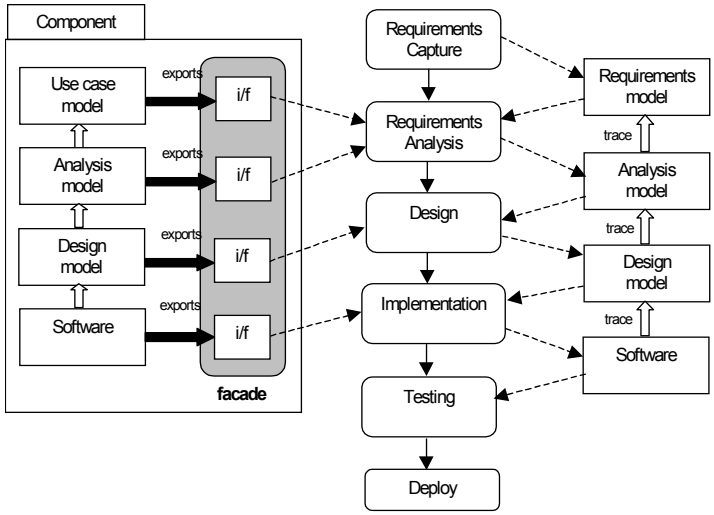
2.1.1.1 REUSABLE COMPONENT MODELLING APPROACH

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. 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 accommodated the reuse of components are only likely to become apparent during the design process, therefore possibly countering aspects of the system's analysis model. 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 2a. 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 there is no commonly accepted framework that supports a suitably wide range of components. The methodological guidelines for component reuse presented here are motivated by the absence of such a framework. As such, the methodology presented in this paper 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 also aimed at supporting reuse decisions based on the 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.



a) Conventional (design model level) component reuse



b) Analysis model level component reuse

Figure 2: Differing Approaches to Component Reuse

The approach is derived from that described in Jacobsen's OOSE methodology [jacobsen97]. 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 2b.

The presentation of a component for reuse in this way is termed 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 usefulness of the facade is strengthened if there is clear traceability between the different models, so that 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. Jacobsen suggests three types of object that may be used in forming a robustness model:

- *Entity objects* that represent long-lived data in the system under analysis.
- *Boundary object* that deal with the interactions between the system and its environment.
- *Control objects* that deal with the dynamic behaviour of the system as described in use cases, and in particular the interactions between boundary and entity objects.

The object types were therefore used as the basis of an enhancement of the component façade concept, referred to here as a *Projection*.

2.1.1.2 OPEN BUSINESS PROCESS MODELLING APPROACH

The attempt to provide a standardised architectural framework for analysing business requirements for SMS have 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 where therefore chosen as the basis for a model that will enable 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 in mapping the TOM to the TINA business model and reference points was to identify which TMF processes operate in which TINA Business Roles. A mapping of the TMF business processes onto TINA business roles, initially presented in [lewis99a], is given in Figure 3.

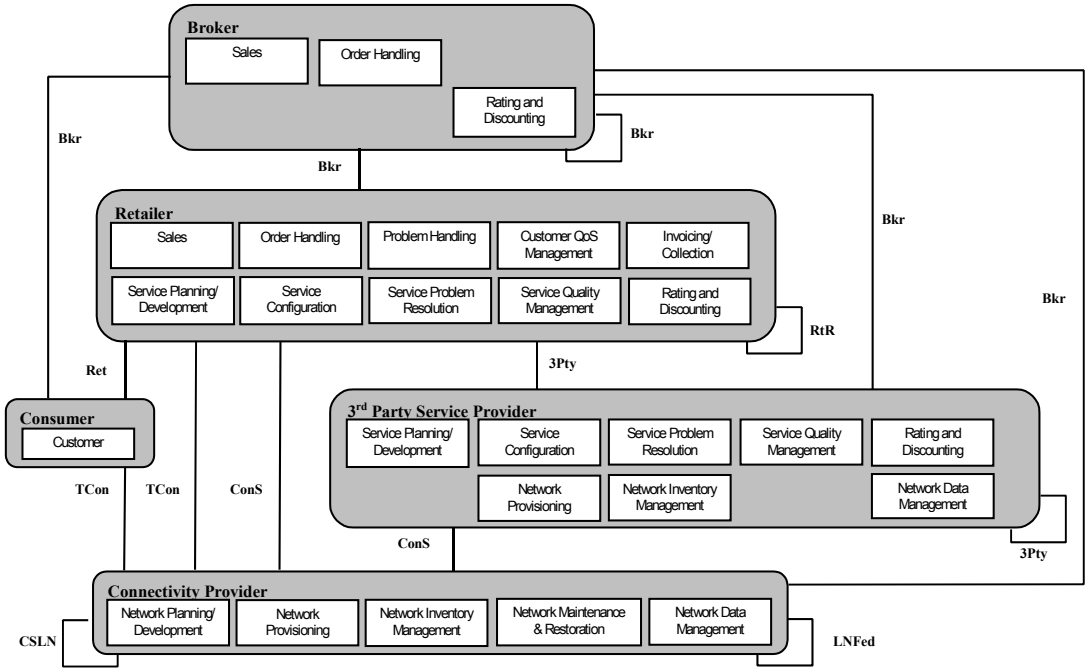


Figure 3: Mapping of TMF Business Processes onto TINA Business Roles

The TOM provides a model of suitable business processes which reflect the typical operations of a service provider. This mapping, therefore, helps in the analysis 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. 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.

3 METHODOLOGICAL GUIDELINES

A series of case studies were conducted around the development of SMS in a number of research projects. These culminated in the FlowThru Project, which developed and validated specific techniques for constructing from reusable component the SMS that implemented specific process information flows. FlowThru provided evidence of the development techniques that developers found most useful in practice. Based on this evidence and the analysis of current development techniques 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's development processes and externally for exchanging models between developers involved in these processes.
- The Projection packaging construct should be used for publishing COTS software, publishing standards and for documenting internally developed reusable software.
- Where possible, an analysis and design process that uses OOSE analysis modelling should be adopted.

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. As recommended above, the core notation used is UML, specifically the OMG's current version 1.1 [ad/97-08-03]. 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 these relationships with existing UML model elements, identified for convenience by the stereotype marker <<uml1>>. Existing 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 *Business Requirements Model* combining Business Process, Business System and Use Case Models
- A *Projection* modelling construct which is a refinement of Jacobsen's facade construct.

3.1 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 already described in UML, 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 4.

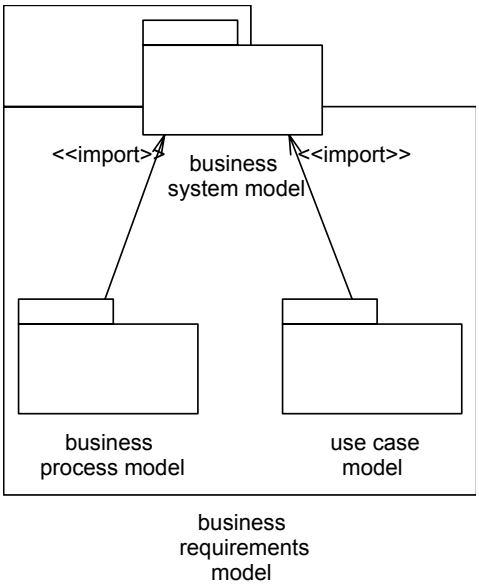


Figure 4: Structure of the Business Requirements Model

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.

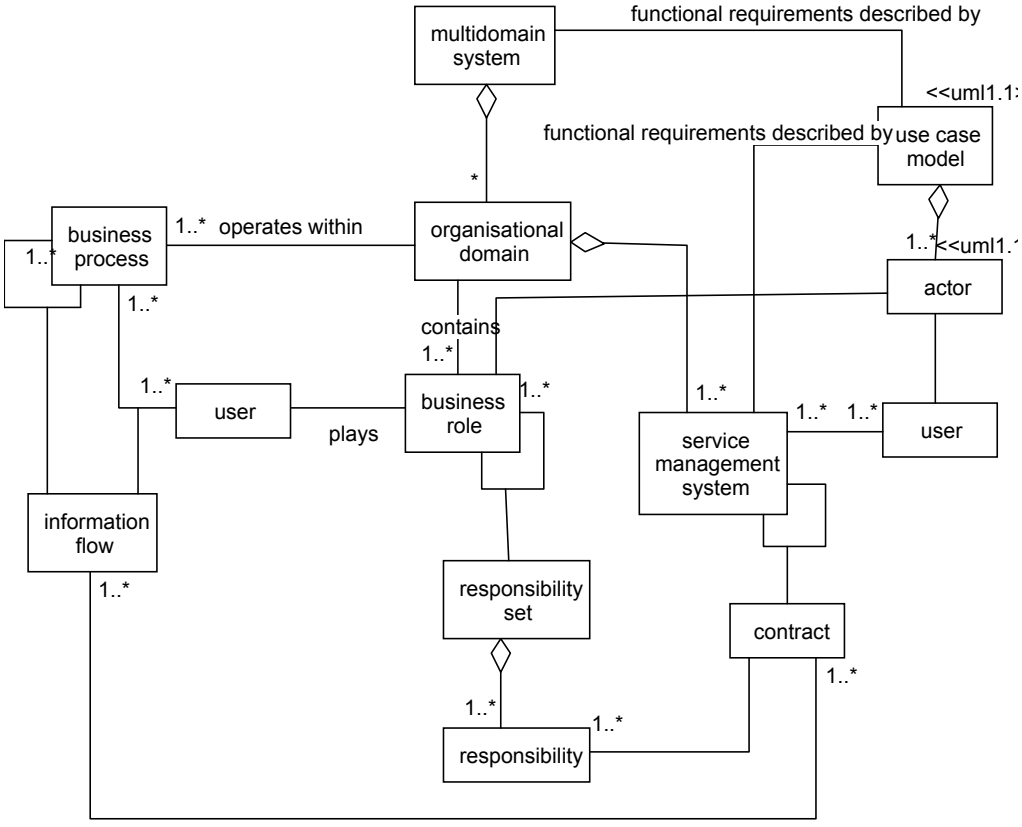


Figure 5: Relationships between the Elements of the Business Requirements Model

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

- Business Roles should map one to one to actors in the use case model, so the descriptions 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 in 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.

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.

Second, establish a use case model where the actors represent the different Business Roles from the multi-domain organisational model and the use cases describe a system consisting potentially of multiple Organisational Domains.

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 6 shows such a diagram for the processes, roles and domains identified in Figure 7 for the fulfilment scenario defined for FlowThru.

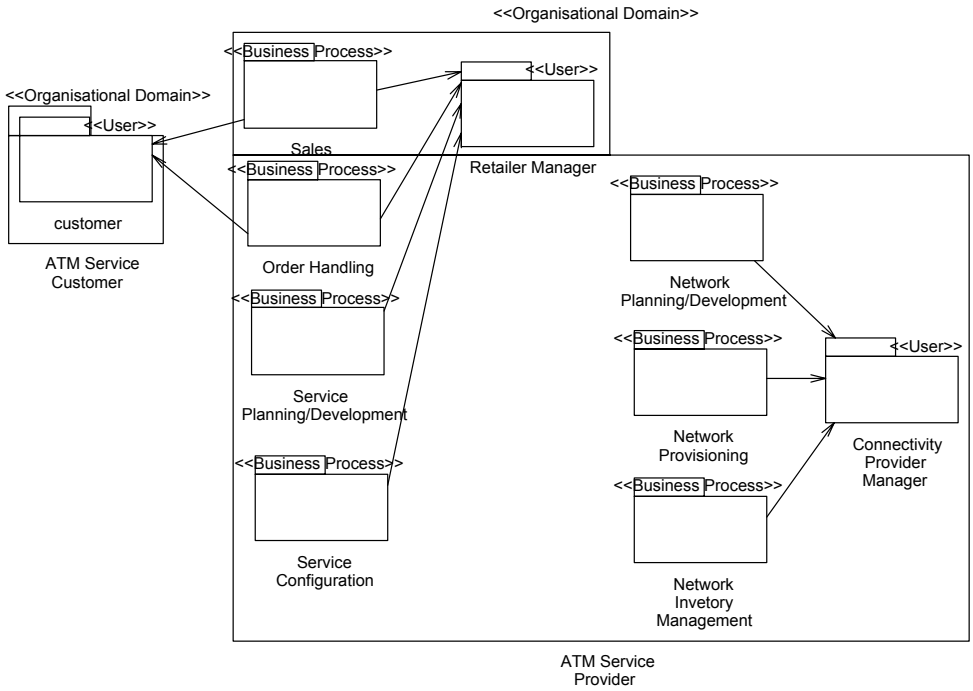


Figure 6: Example of static Business Process Model using a UML Component Diagram

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 7 for the processes and users in Figure 6, for the "subscribe to ATM service" use case.

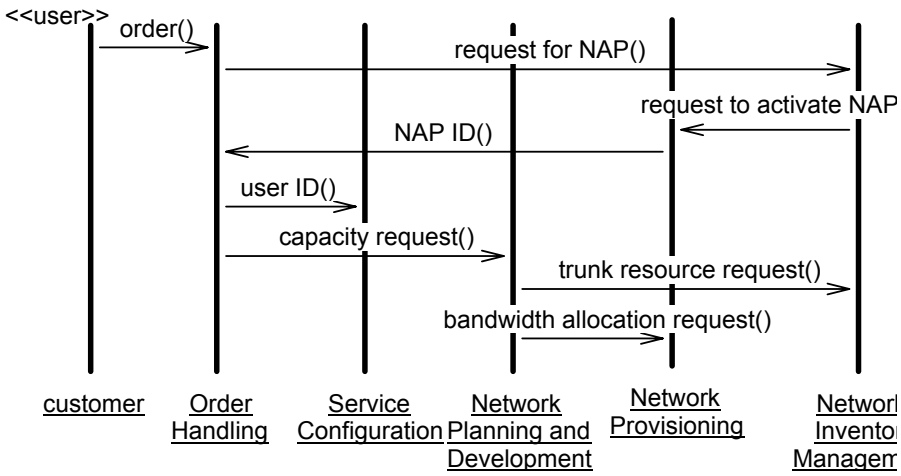


Figure 7: Example of Dynamic Business Process Model using a UML Sequence Diagram

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 8 for the SMS making up the fulfilment trial business system.

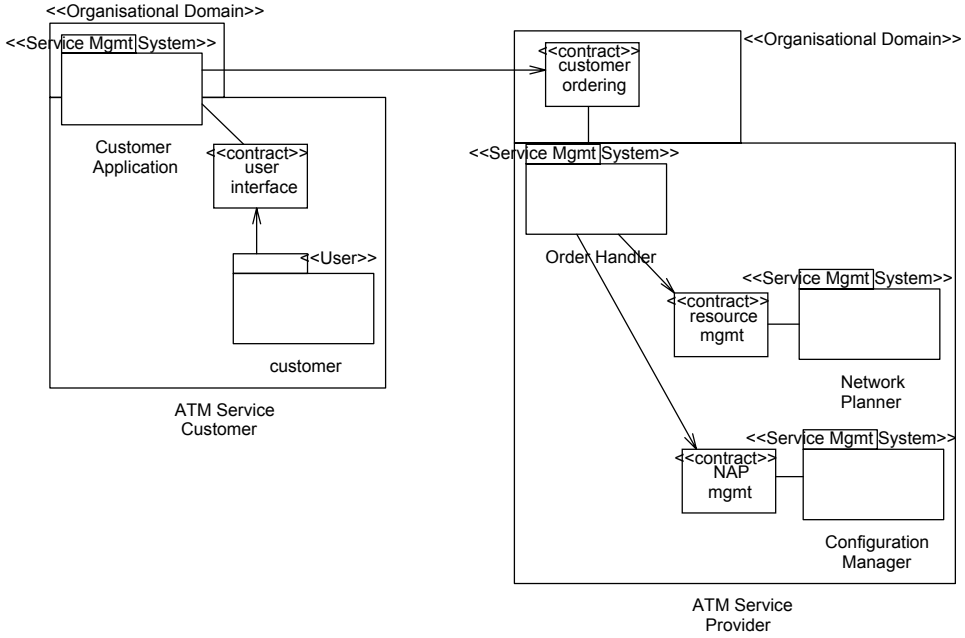


Figure 8: Example of SMS Level Business System Model using a UML Component Diagram

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.

3.2 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. 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 façade construct currently defined in UML. This structure is shown in Figure 9.

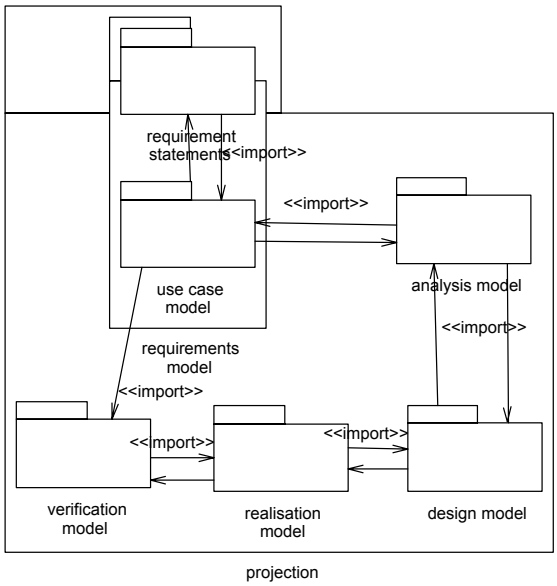


Figure 9: Structure of the Projection Modelling Construct

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.

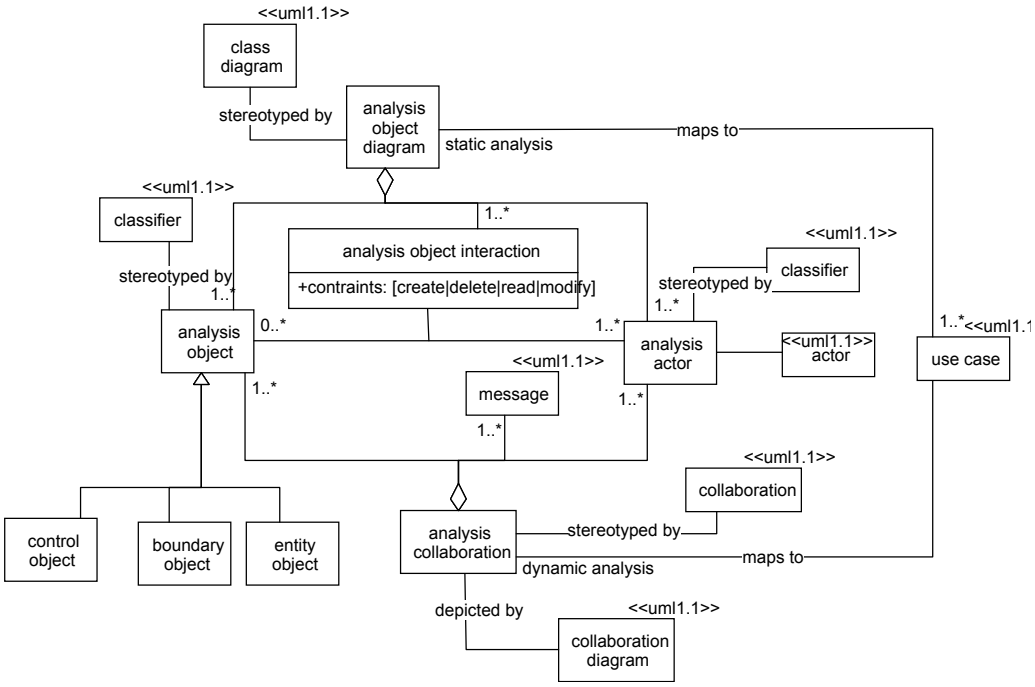


Figure 10: Relationship between Elements of the Projection Construct's Use Case and Analysis models

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 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 10. An example of an Analysis object Diagram is given in Figure 11.

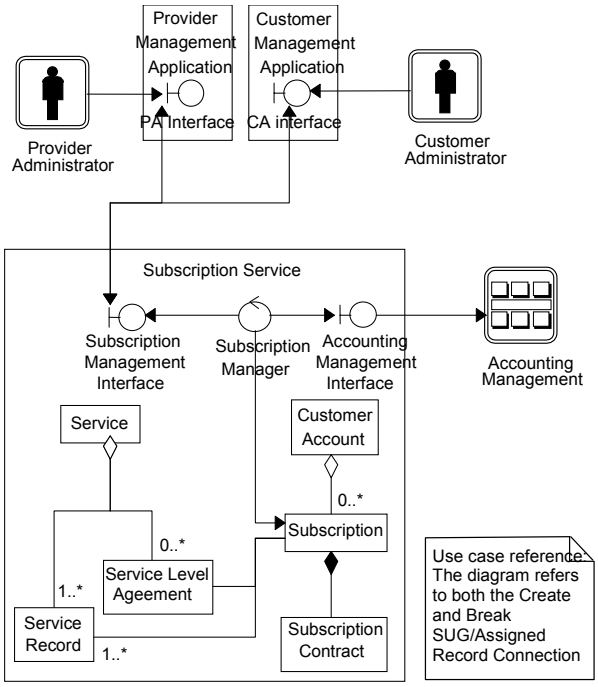


Figure 11: Analysis Object Diagram for Subscribe a Customer to a Service Use Case

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 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.

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.

4 APPLICATION OF THE GUIDELINES

The Projection modelling approach was applied within FlowThru to the integration of components in three separate scenarios demonstrating different business process areas from the TMF's Telecoms Operations Map. The components from which these systems are constructed were 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 Projections. The core aim was to be able to export the Projection 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 publication of the facade on the web is recounted in more detail in [lewis99b].

In mapping use cases to an analysis model for a facade, entity objects were derived from noun phrases that occur in one or more use cases. Boundary objects were identified by any interactions between the users and the system. Control objects were initially allocated per use case, and then consolidated as functional commonalties are identified between use cases. Interaction diagrams were 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 Projection took into account design level issues, such as scalability, load balancing, platform dependencies, database schemes etc. Strong tracing between objects in the analysis model and the design model had to be maintained during this process.

Though it is preferable to generate a Projection 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 the use of Projections is that that they can hide the internal model if necessary. In FlowThru the latter situation was demonstrated by developing Projections for components that had been developed and documented using ODP viewpoints. The following mappings were be used to help reverse engineering the ODP models into the Projection format:

- Roles in the enterprise viewpoint onto actors in the use case modelled in the Projection.
- Computational objects from the computational viewpoint onto control objects in the Analysis Model.
- Computational object interfaces, individually or grouped, onto boundary objects in the Analysis Model.
- Information object onto entity objects in the Projection's Analysis Model.
- Where sequence diagrams had been used to clarify the interactions between computational and information objects, then these formed the basis for reverse engineering use cases, otherwise use cases were based on the component's requirements.

The analysis of a management task in a specific business scenario was 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 were then broken down into internal activities performed with the system, using a UML activity diagram. The activities in these diagrams were placed within swim-lanes representing TMF business processes, were necessary residing in different administrative domains. This eased the identification of which existing TMF business agreements matched the requirements of the task at hand. The mapping of the TOM process onto the TINA business roles also then enabled the identification of where TINA reference point definitions should apply. The activity diagram for the "Subscribe to ATM Service" use case is presented in Figure 12.

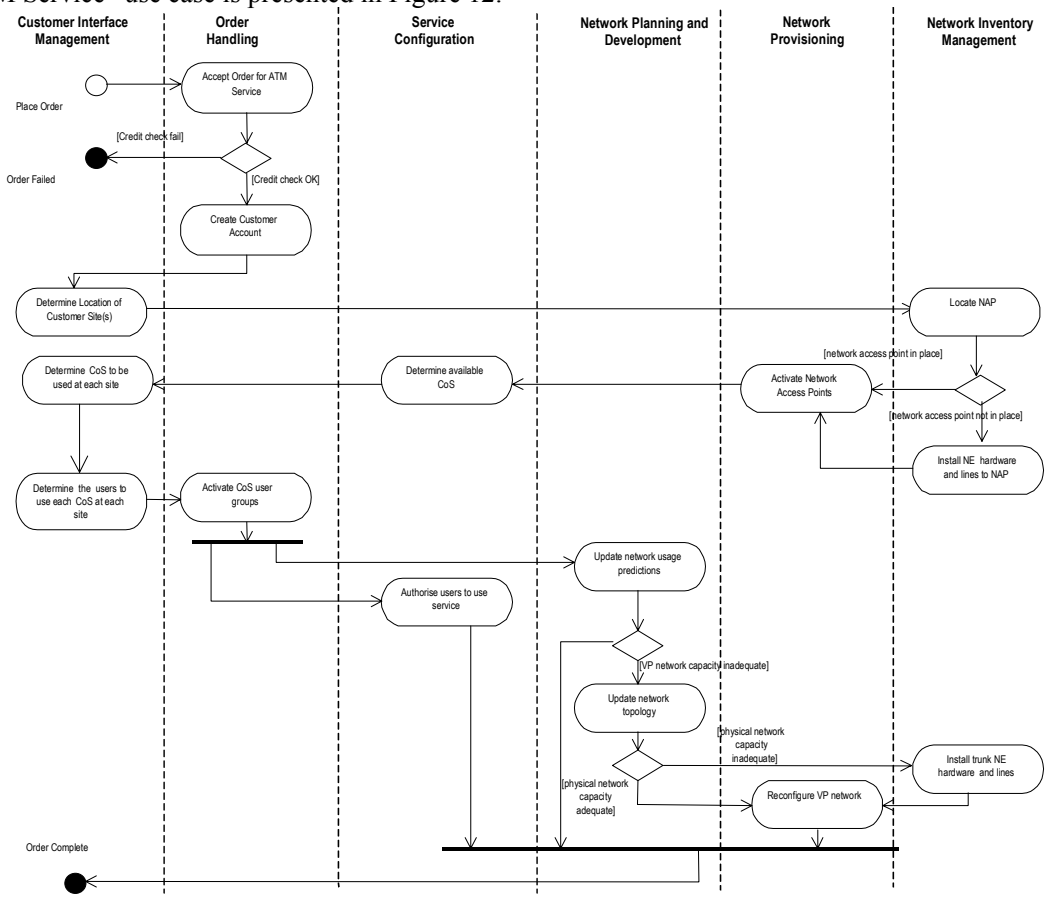


Figure 12: UML Activity Diagram for the Subscribe to ATM Service Use Case

By comparing these activities to the use cases in the imported components' Projections, 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 12 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 then defined in terms of the interactions between analysis

model elements from the imported component Projections, 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.

4.1 EVALUATION OF APPROACH

The experiences of this case study relate the generation of the Projection for pre-existing components and to 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 Projection presents the component model at levels of abstraction, i.e. as Use Case and Analysis Models, which facilitate the components integration into the SMS. It was found that the division in the Projection 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 Projection hid all detailed design issues except those relating to the interfaces via which it is re-used.

Publishing the Projection 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. However, the fact that the same teams were involved in Projection generation and the design of the systems that reused the components meant that the effectiveness of the Projection construct in component selection and comprehension could not be assessed. In addition, therefore, a questionnaire was used to get a more structure view of the developers' experiences, against which the above observations may be compared. Fourteen completed questionnaires were returned. The responses mostly were in line with the above observations and the findings of the previous case studies. The business process modelling was rated highly for both system and component development. The Projections were also rated highly by system designers, though the rating for the Projection's Use Case Model, indicated that this was not clearly related to the Analysis and Design Model in all cases.

The responses to some open questions included in the questionnaire were as follows:

- In response to the question "Do you think the use of component Projections 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 Projection 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 Projection 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 Projections?" five replied yes, none no and nine did not express an opinion.

5 CONCLUSIONS

This work has provided experience in the application of UML and commercially available CASE tools for service management process analysis and component reuse. It is hoped these results will prove useful to industry practitioners faced with building service management systems. The methodology and experiences presented here are currently being disseminate via the ACTS program as well as providing input to the TMF, TINA and ITU workgroups.

The business process to 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 based 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 Projections 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.

6 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.

7 REFERENCES

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

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

[nmf-504] SMART Ordering - SP to SP Interface Business Agreement, NMF 504, Issue 1.0, TMF, Sep 1997

[nmf-gb910] NMF Telecoms Operation Map: A high-level view of end-to-end service fulfilment, service assurance and billing, NMF, Morristown, 1998

[vincent] Modeling/Design Methodology and Template, Draft 4, Vincent, A, Hall, C., TMF, Oct 1997

[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

[lewis95] Experiences in Multi-Domain Management Service Development, Lewis, D., Tiropanis, T., Bjerring, L.H., Hall, J., in [ISN95], pp174-184, Springer-Verlag, Oct 1995

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

[lewis99b] Modelling Management Components for Reuse Using UML, Lewis, D., Malbon, C., DaCruz, A., Proceedings of the 6th International Conference on Intelligence in Services and Networks, Barcelona, Spain, pp210-222, Springer-Verlag, Apr 1999

[lews99c] 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