The Development of Integrated Inter and Intra Domain Management Services

D. Lewis,
Department of Computer Science,
University College London,
United Kingdom.
D.Lewis@cs.ucl.ac.uk

V. Wade,
Department of Computer Science,
Trinity College Dublin,
Ireland.
Vincent.Wade@cs.tcd.ie

R. Bracht,
IBM,
Heidelberg,
Germany.
bracht@de.ibm.com

Abstract

Service providers are under increasing competitive pressure to reduce costs, increase customer service and introduce new services. Speed of deployment, ease of reuse of existing components and co-operation across organizational boundaries are vital for management systems to succeed in such an environment. This requires a significant change in the way management systems are modeled and developed. This paper presents a design process which is customized to support the analysis and development of inter and intra domain management systems as well as assisting in the development and deployment of reusable components. The design process was trialed and validated by its application in the development of software in a multi-domain management research project.

Keywords

Service Management, Software Methodologies, Open Service Market, Component Reuse

1. Introduction

The liberalization of the telecommunications market world-wide and the advent of increasingly intense competition is forcing service providers to reduce the cost of delivering services while increasing customer satisfaction. The automation of operations systems is seen as an important part of achieving this, improving the efficiency of the interactions between a service provider's business various processes and the interactions with the processes of customers, suppliers and collaborators [1]. As the pressure intensifies to rapidly develop effective telecommunications management systems, developers are increasingly turning to *software reuse* techniques and in particular the use of commercial off the shelf

components. In addition, increasing emphasis is being placed on the use of *open interfaces* in order to support inter-domain interfaces that ease multi-domain interworking problems and conform to regulatory requirements.

An appreciation of the overall business environment in which open telecommunications management systems are developed can help in our understanding of the requirements for a suitable development methodology. A model of this environment is depicted in Figure 1. It centers 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 for common management functions implemented by components. 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.

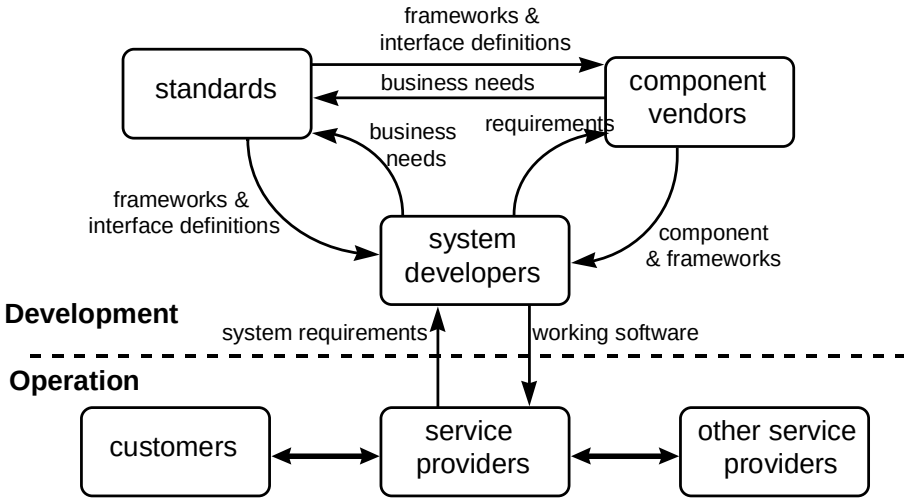

Figure 1: Stakeholders in the management software development process

Any suitable development methodology must have both a suitable notation for expressing the various concerns that arise during system development, as well as a process for systematically addressing those concerns from initial requirements capture through to testing and deployment of software. Such a development methodology is presented in this paper. It builds on current management software

paradigms and general software engineering trends and has been validated by application in the development of substantial multi-domain management systems for advanced telecommunications services. The paper first details the work that has influenced the definition of this methodology, before giving an overview of the methodology itself and providing some detailed examples of its application to the development of research prototypes.

2. Relevant Trends in Systems Development

The ITU-T's Telecommunication Management Network (TMN) recommendations [2] provide an architectural framework for constructing open telecommunications management systems. The series includes a TMN Interface Specification Methodology [3] that provides guidelines for the functional decomposition of management functions resulting in agent interface specifications expressed in GDMO. It does not provide specific guidance on the development of entities with multiple manager and agent interfaces, which will typically be key to inter-domain management interactions.

There has also been interest recently in applying Open Distributed Process (ODP) techniques to management system development. The OSI ODP standards [4] have been applied to telecommunication management by the TINA consortium [5] and ISO's Open Distributed Management Architecture work [6]. Though ODP provides a structured way of modeling distributed system concerns using separate Enterprise, Information, Computation, Engineering and Technology viewpoints, it does not provide a prescriptive development process, and consistent notations for all the viewpoints have not yet been standardized.

The broader software engineering community has developed a wide range of methodologies, aimed more at the problem of software development than open interface definition. Though no consensus has emerged on the ideal development process, this being seen as specific to the application domain, there is now a standard for the graphical representation of object-oriented analysis and design models. This has been standardized by the Open Management Group as the Unified Modeling Language (UML) [7]. The methodology presented in the next section uses UML in a process tailored to developing open telecommunications management systems.

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 [8], which has also used UML. The case studies provided examples of the application of UML use case diagrams for capturing the requirements of management systems, and UML class, sequence and component diagrams for the design of these systems. The ACTS project TRUMPET performed a case study of an inter-domain service management problem that used ODP viewpoints modeled using UML [9]. 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. Some problems were identified however with UML's ability to represent ODP computational objects.

3. A Development Methodology

In order to develop management systems that are able to satisfy the requirements both of the organization that operates them and of the multi-organization collaborations that the open service market may demand, two model types must be accommodated:

- *Multi-Domain Model*: This captures requirements of management tasks involving more than one organizational domain. It therefore concentrates on supporting *inter-domain* interactions.
- *Single-Domain Model*: This captures the management system requirements and design of a specific organization. It therefore concentrates on *intra-domain* interactions.

However, these models are not independent, and a methodology for this problem domain has to support aligning requirements and interface definitions from the multi-domain model and the single domain model.

In addition to these two models a suitable methodology must also specifically address the modeling of components. Components should be treated as separate entities from multi-domain or single-domain systems, and if developed by third party vendors, they will have distinct development life-cycles. The methodology must therefore support the inclusion of a *component model* into the development of multi-domain or single-domain systems. This methodology has been influenced in making these distinctions by the modeling work of previous telecommunications research projects and programs such as PREPARE [10][11] and PRISM [12] and TINA-C's modeling guidelines [13].

UML has been selected as the primary modeling notation for the methodology due to the broad range of models it supports, its extensibility though its stereotype mechanism and its increasing support by CASE tools. However, to fully support the development cycle of management systems, detailed design specifications and open interface definitions have to be expressed in more technology specific languages, such as GDMO, SMI and IDL. Tools are already available to support mappings from UML to these languages and such mappings form the basis for UML modeling in support of ongoing standardization work in the TeleManagement Forum.

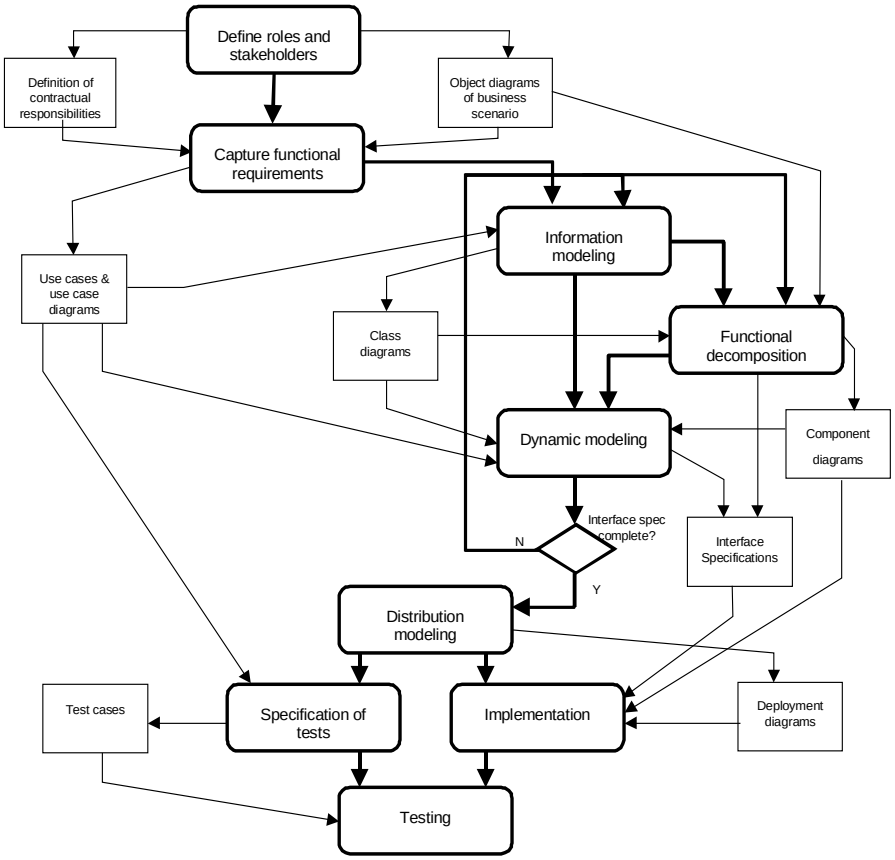

Figure 2: Overview of Development Process

The development process aims to provide a structured way of iterating through the development of management systems, whether they are multi-domain systems, single-domain systems or components. By following a common well understood process and notation, regardless of the type of system being implemented, communication between developers of these different types of systems will be facilitated. The process can be decomposed into the following steps:

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

- Definition of test specifications.
- Implementation and integration of components.
- Testing of sub-systems, sub-system integration testing and testing of external interactions.

This process is represented in Figure 2, together with the modeling outputs of the individual steps. Note that some models are used in several subsequent steps. Also note that the information modeling, functional decomposition and dynamic modeling steps are closely coupled and may undergo several iterations before arriving at a set of completed interface specifications.

4. Application of Methodology

The methodology outlined above was applied in the EU sponsored ACTS project called Prospect. The target multi-domain systems developed in Prospect were for performing user trials to demonstrate and assess the integration of service control systems with service and network management systems, operating across a number of organizations. Several trial systems were developed each corresponding to a different multi-domain system. The different trials demonstrated how systems could be constructed to flexibly support multiple business scenarios and how management components could be reused in different domain systems, across these different business scenarios. System development was based as much as possible on the use of UML, using both the Paradigm Plus and Rational Rose CASE tools. The following sections presents examples of the models generated during the definition of role, stakeholders and use cases of multi-domain systems and during the information modeling, functional decomposition and dynamic modeling of components.

4.1 Multi-domain Modeling

In any multi-domain scenario, such as the Prospect trials, a clear model is required of the different stakeholders that are involved and the business roles supported by these stakeholders. The enterprise model for a specific business scenario was represented using UML object diagrams. The objects in the enterprise model diagrams were instances of classes from a general enterprise model, shown in the class diagram in Figure 3. In this general model, roles and stakeholders are differentiated by their class stereotypes. The general enterprise model defined a set of roles and stakeholder that were thought likely to be present in multi-domain, open service management scenarios. However, this was principally performed to clarify the context of Prospect's work, and other general enterprise model classes could be equally valid in different situations.

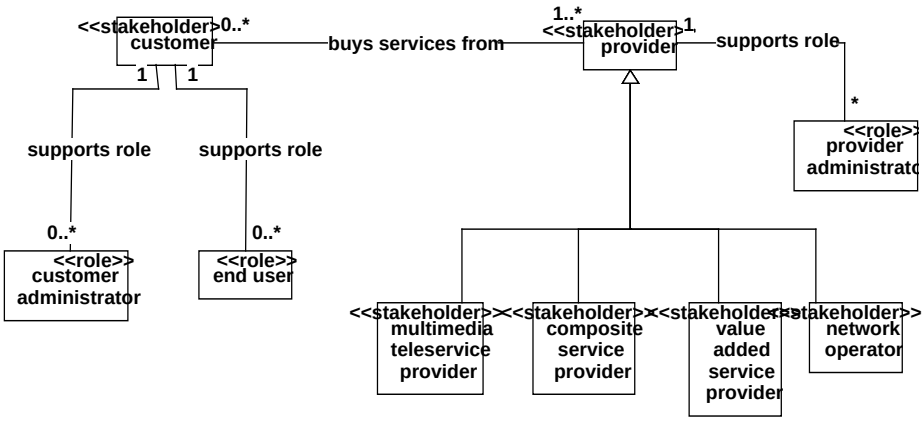

Figure 3: Class diagram showing Roles and Stakeholders used in Prospect Trials

To provide a more detailed context for the subsequent definition of use cases, the relationship between the roles and organizations prior to the trial was also described. This took the form of statements of contractual responsibilities between the different stakeholders that could, in commercial scenarios, form the basis of or be informed by legal contracts between the parties concerned. These contractual responsibilities may be represented as associations between stakeholder objects.

Use cases at the multi-domain system level define what the system as a whole needed to do in terms of useful interactions with actors that define the system's environment. These actors represented instances of the role class stereotypes from the general enterprise model for the multi-domain system. The use cases themselves were stated in the form of text, with sections defining the use case pre-conditions, the use case itself and the use case post conditions. The preconditions present the state of the multi-domain system, from the point of view of the actors, and would typically be related to the post-condition of other use cases. The use case body describes in terms meaningful to the actor, the interactions that they performed with the system in order to perform some useful task.

To analyze the inter-domain interactions within such a multi-domain system, the individual use cases were refined to describe the inter-domain interactions they required. This step was informed by the responsibilities between different roles and stakeholders in the enterprise model. Refining the multi-domain use cases in this way enabled the identification of use cases for the individual domain, i.e. the single-domain systems that made up the multi-domain system. A similar set of use case diagrams showing the decomposed, single-domain use cases, could them be produced. However, use case diagrams in UML do not support direct interaction between use cases, so use case diagrams could not be used to show the full chains of interactions between single-domain systems. Instead, high-level sequence

diagrams were also to show information flows between the multi-domain system actors and the single-domain systems.
The definition of what actual information was involved in these inter-domain use cases was based on the requirements embodied in the multi-domain use case definitions. However, developers also took information definitions from existing standards and from existing components that were likely to be used in the systems implementation.

4.2 Component Modeling

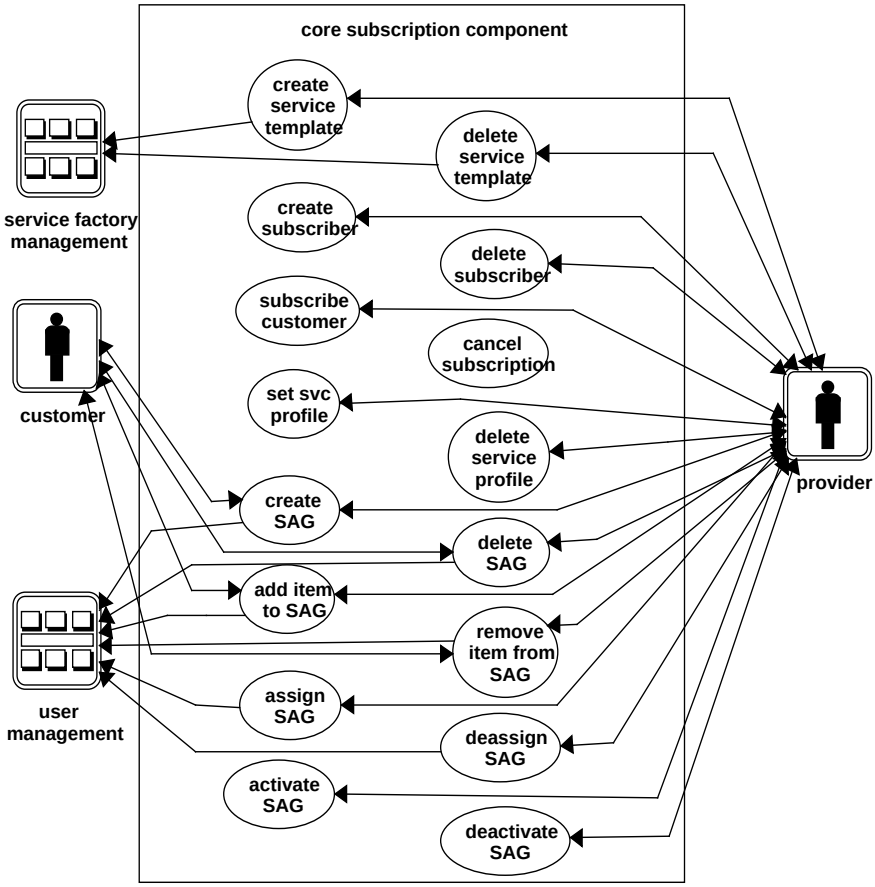

Figure 4: Use case model for Prospect Subscription Management component

The components used in Prospect were mostly taken from the TINA Service Architecture [14], where they were described in terms of ODP viewpoints using OMT for the informational viewpoint [15] and block/interface diagrams and ODL (an extension of IDL) for the computational viewpoint [16]. This was found to be

inadequate for understanding the component as a whole during system analysis, since it represented the component at a design level only. Higher level views of the component were only available from TINA in terms of unstructured text and diagrams together with some examples.
Components are in themselves systems, and can therefore benefit from the same modeling approach used for systems. Components were therefore re-described using use cases to define the actors that would interact with a component and to define those interactions. A use case diagram for the Subscription Management component used in Prospect is given in Figure 4. Note that some actors represent human users while others represent systems that may be other components in the same family.

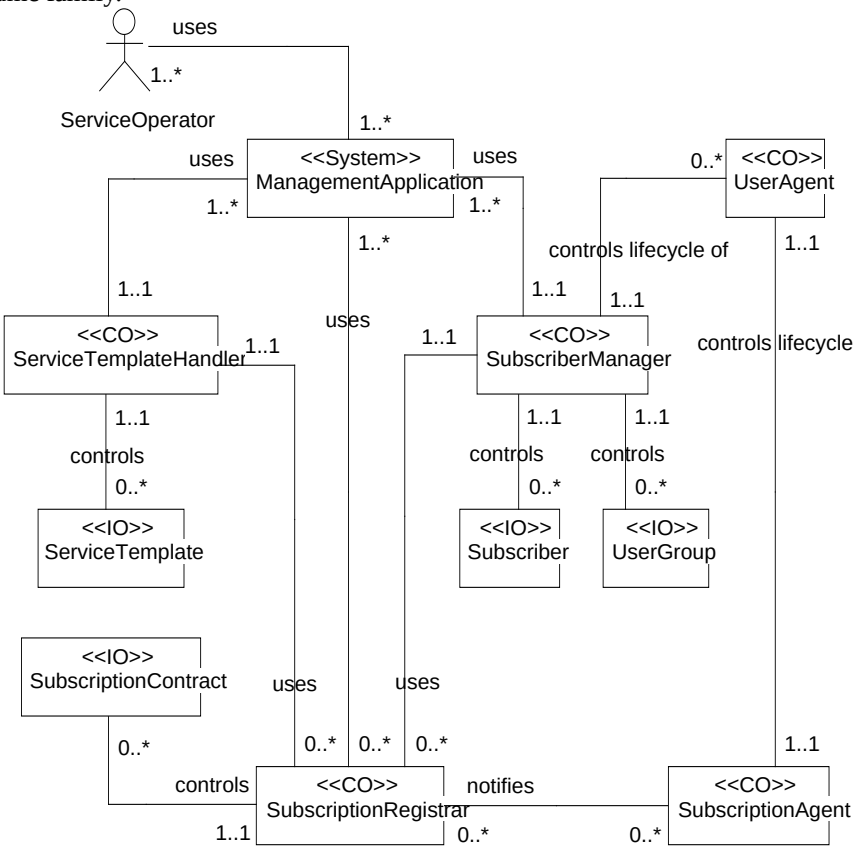

Figure 5: Top level class diagram for Subscription Management component design

The design of components was developed using UML class diagrams. These represented the results of both the information modeling activity and the

functional decomposition activity. Outputs specific to either activity were integrated on the same diagrams but were differentiated by stereotypes for information objects (IO) and computational objects (CO) respectively. Though UML component diagrams could be used to identify the different interfaces of computational objects, the details of the interfaces were modeled by refining the class diagrams. This facilitated both the automated generation of IDL by case tools and the maintenance of consistency with interaction diagrams, features not directly supported by component diagrams. Figure 5 shows the top-level class diagram for the Subscription Management component.

Figure 6 shows an example of how class diagrams were used to define the interfaces for one of the computational objects that makes up the Subscription Management component shown in figure 5. The computational object, Subscriber Management, has eight IDL interfaces, two of which have been inherited from more general interfaces intended for managing a computational object's lifecycle (i_CoInit) and administrative state (i_CoMgmt). The interface operation's parameters are not shown in this figure, but were also modeled in the CASE tool used (Rational Rose in this case).

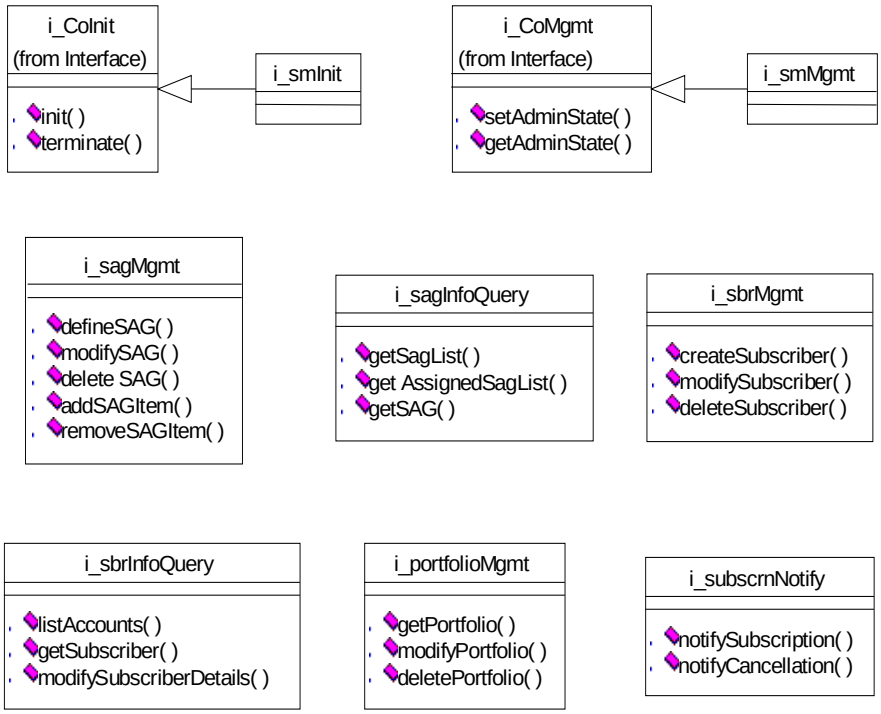

Figure 6: UML class diagrams showing details of the interfaces to the Subscriber Management Object

To fully describe the component behavior, its dynamic operation had to be modeled. This was typically performed by defining the interactions between the computational objects and actor systems based on individual use case descriptions. An example of such an interaction diagram is given in Figure 7, which shows the interactions between entities in terms of IDL operations on their interfaces. This example performs the functionality required by the "Create SAG (Subscription Assignment Group)" use case shown in Figure 4. Note that in this diagram only the Subscriber Manager and the Subscription Agent entities are part of the Subscription Management component. The management application is the design level representation of the application used by the Provider Administrator actor identified in the use case model, while the User Agent object is part of the User Management system actor also identified in the use case model.

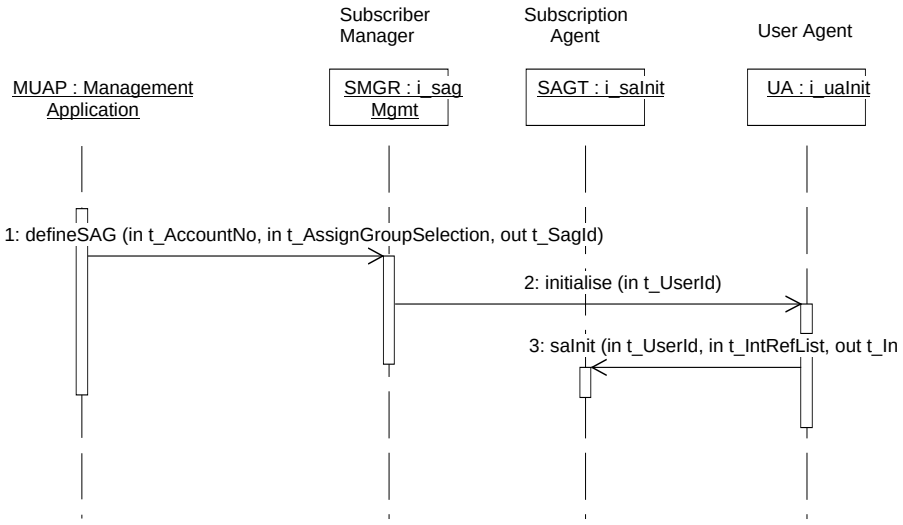

Figure 7: Interaction Diagrams showing subscription Component behavior for the Create SAG use case.

5. Conclusions and Further Work

A development methodology has been proposed that combines an iterative, use case driven development process with UML as the modeling notation. This has been applied effectively in the Prospect project for multi-domain management system analysis, intra-domain system development and component development. By using the same basic methodology for all of these activities communication between the different developers, e.g. component developers and component re-users, was eased. This methodology has been adopted as the basis for an ACTS Guideline on designing service management systems [17]. This experience also provides evidence that open management system development can be performed

using existing software development techniques as already advocated within the TeleManagement Forum [18].

Though the use of UML is key to this common approach some difficulties in its application were encountered. Use cases lack a relationship where use cases in one system interact with use cases in another system, a facility that would be useful in decomposing multi-domain system requirements into single-domain system ones, and similarly in matching single domain system requirements onto component use cases. Also, though multi-interface computational objects can be modeled as components in UML component diagrams, these cannot be used as communicating entities in interaction diagrams, restricting support for multi-interface distributed components.

Further work is underway within the ACTS project FlowThru in refining this methodology. This is principally aimed at providing better support for analyzing business process requirements using UML activity diagrams, and at ensuring component specifications are restricted to providing only the details actually needed for reuse.

Acknowledgment

This work was conducted under the partial funding of the EU through the ACTS projects Prospect (contract AC053) and FlowThru (contract AC335). The views expressed in this document do not necessarily reflect those of these consortia.

References

[1] E Adams, K Willetts, *The Lean Communications Provider*, McGraw Hill, 1996.

[2] *Telecommunication Management Network*, M3000, ITU-T, 1996.

[3] *TMN Interface Specification Methodology*, ITU-T Draft Revised Recommendation M.3020, 1994.

[4] *Reference Model for Open Distributed Processing*, Part 1 Overview and Part 2 Foundations, ISO/IEC 10746-1 (DIS) and 10746-2 (IS) ITU-T X901 and X902, 1994.

[5] M.Chapman, S. Montesi, *Overall Concepts and Principles of TINA*, TINA Baseline Document, TB_MDC.018_1.0_94, December 1994.

[6] Information Technology- Open Systems Interconnection- *Open Distributed Management Architecture*, ISO/IEC Draft International standard, Draft Recommendation X.708, June 1996.

[7] *Unified Modeling Language Specification*, v1.1, OMG, August 1998.

[8] F. Nesbitt, T. Counihan, J. Hickie, *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.

[9] M. Kande, S. Mazaher, O. Prnjat, L. Sack, M. Wittig, *Applying UML to Design an Inter-Domain Service Management Application*, Proceeding of UML'98, Mulhouse, France, June 1998.

[10] J. Hall (ed), *Modeling and Implementing TMN based multi domain management*, PREPARE Consortium, Springer Verlag, 1996.

[11] D. Lewis, T. Tiropanis, L.H. Bjerring, J. Hall, *Experiences in Multi-domain Management Service Development*, Proceedings of the 3rd International Conference of Intelligence in Broadband Services and Networks, Heraklion, Greece, Springer Verlag, October 1995.

[12] A. Berquist, *Succeeding in Managing Information Highways*, PRISM Consortium, Springer Verlang, 1996.

[13] J.Salleros, *TINA-C Service Design Guidelines*, TINA Report TP_JS_001_0.1_95, TINA Consortium, March 1995.

[14] H. Berndt, R. Minerva, *TINA-C Service Architecture*, TINA Baseline document TB_MDC.018_1.0_94, 1994.

[15] H. Christensen, E. Colban, *Information Modeling Concepts*, TINA Baseline document TB_EAC.001_1.2_94, Version 2.0, April 1995.

[16] *Computational Modeling Concepts*, TINA Baseline document TB_A2.NAT.002_3.0_93, December 1993.

[17] V. Wade, D. Lewis, W. Donnelly, D. Ranc, N. Karatzas, M. Wittig, S. Rao, *A Design Process for the Development of Multi-domain Service Management Systems*, ACTS Guidelines for ATM deployment and interoperability, Baltzer, June 1997.

[18] A. Vincent, C. Hall, *Modelling/Design Methodology and Template, NMF Internal Document*, Draft 4, 18th October 1997.

Biographies

David Lewis graduated in Electronic Engineering at the University of Southampton in 1987 and in 1990 received a MSc. in Computer Science from University College London where since he has worked as a research fellow. He has worked primarily on the EU funded projects in which he has been responsible for leading teams developing integrated, multi-domain service management

systems. He is also working on a Ph.D., researching a service management development framework for the open services market.

Vincent Wade is a lecturer in the Computer Science Department in Trinity College Dublin. He received his BSc from University College Dublin, Ireland and a MSc from Trinity College Dublin, Ireland. He leads a research group investigating distributed information systems, management systems and virtual environments. He currently leads several EU and industrial research project in these areas and is author of over forty technical papers in international conference and research journals.

Ralf Bracht received a Ph.D. in physics from Ruprecht-Karls University, Heidelberg, Germany, in 1995. Since then he has been working at IBM Science and Technology Center in Heidelberg. He has worked on the RACE project PREPARE and the ACTS project Prospect. He was Prospect's technical chairman and is now a project manager at IBM. His current interest is management of telecommunications services.