A Review of Approaches to Developing Service Management Systems

David Lewis
Department of Computer Science
University College London,
E-Mail: D.Lewis@cs.ucl.ac.uk

Abstract:

As service management systems are deployed in an open service market environment, the pressures on their developers to shorten development times and improve the flexibility of systems will increase. Component reuse may help in alleviating some of these pressures; however, while object-oriented reuse is used in the definition of interfaces for management systems, the systematic application of reuse to the systems' internal design is not well established. This paper examines some of the approaches applicable to service management system development from the Telecommunications, Distributed Systems, Internet and Software Engineering fields. Experience in applying some of these approaches is presented, and based on this, refined approaches to service management system development are discussed.

Keywords:

Service Management, Open Distributed Systems, Object-oriented Development Methodology, Component Reuse

1. INTRODUCTION

Telecommunications management has traditionally been centered on the management of networks. Standards development in this sector has been driven primarily by the need for open management interfaces to network elements. However, the spread of liberalization and competition through the global telecommunications industry is highlighting the importance of rapidly delivering new, improved services to the customer, at lower cost [1]. In the emerging open market in telecommunications services, therefore, a greater emphasis is being placed on *service management*. This paper examines existing development approaches to that can be applied to service management systems, and assesses them based on development experiences.

The following section details the requirements imposed by the open service market. Section 3 reviews existing approaches from the Telecommunications, Distributed Systems, Software Engineering and Internet/WWW fields. Section 4 assesses relevant experience from recent research projects performing service management system development. Section 5 discusses how these approaches can be consolidated and how further improvements can be made.

2. REQUIREMENTS ON THE DEVELOPMENT PROCESS

Service management aims to ensure that telecommunications services are delivered to the customer with reduced cost and with an excellent level of service, while also enabling the rapid introduction of new services. The open services market will place many additional requirements on service management systems in comparison to those imposed by the state-owned monopoly-based system of service provision. The emergence of a large and diverse population of service providers in the telecommunications market, together with the impact of some aspects of competitive regulation, means that it is very important for a provider's service management systems to interact with those of other providers. If open competition is to be supported through these interactions, service management systems need to conform to inter-domain interfaces specified as open standards or industry agreements of some form. Similarly, to avoid service management systems causing customer lock-in, interfaces between the customer and the provider of services should also conform to open interface definitions.

A further significant requirement on telecommunication management systems, when compared to other aspects of communications software, e.g., signaling and protocol design, is the emphasis on user requirements. Management systems accomplish their role principally by delivering accurate and timely information to human administrators, be they network administrators, help-desk operators, or the customers themselves. Therefore, user requirements play a significant role in the analysis and design of management systems.

The pace of change in network management systems is largely tied to the pace of change in network technology. In a highly competitive services market, the requirements on service management systems will be greatly influenced by factors related to marketing, pricing, responding to competitors' actions, market realignments, e.g., mergers and acquisitions, and the introduction of new services. As these factors are likely to change much more rapidly and unpredictably than the underlying network technology, service management system must exhibit a much higher level of flexibility and robustness to change than network management systems.

The requirement for flexibility and robustness, together with the need for rapid service development motivate the need for reuse techniques in service management system development. Reuse may take several different forms. Reuse of specification involves the use of commonly agreed interface specifications for a component. Reuse of implementation covers the reuse of existing implementations as they are provided, i.e., as a black box without any modification. Reuse of design covers situations where existing designs, possibly with accompanying implementations, exist, but typically need to be modified somewhat to suit a specific application.

Adoption of reuse techniques in software development requires a degree of long term planning since the investment made in identifying, designing, implementing and maintaining a reusable component will only be returned over several instances of its reuse. Reusable software products usually take the form of libraries or frameworks that provide potentially widely applicable functionality, in order to maximize the return through sales on the initial investment. Conformance of a reusable product to an international standard or industry agreement will lessen the risk perceived in developing the product. As many interface specifications have already been standardized for telecommunications management, reusable components addressing this area need to exhibit a tight coupling between specification reuse and implementation reuse. Any methodological appraoch to developing reusable software for telecommunications management has to address the interaction between stqandards developers, component developers and system developers. These relationships are outlined in figure 1.

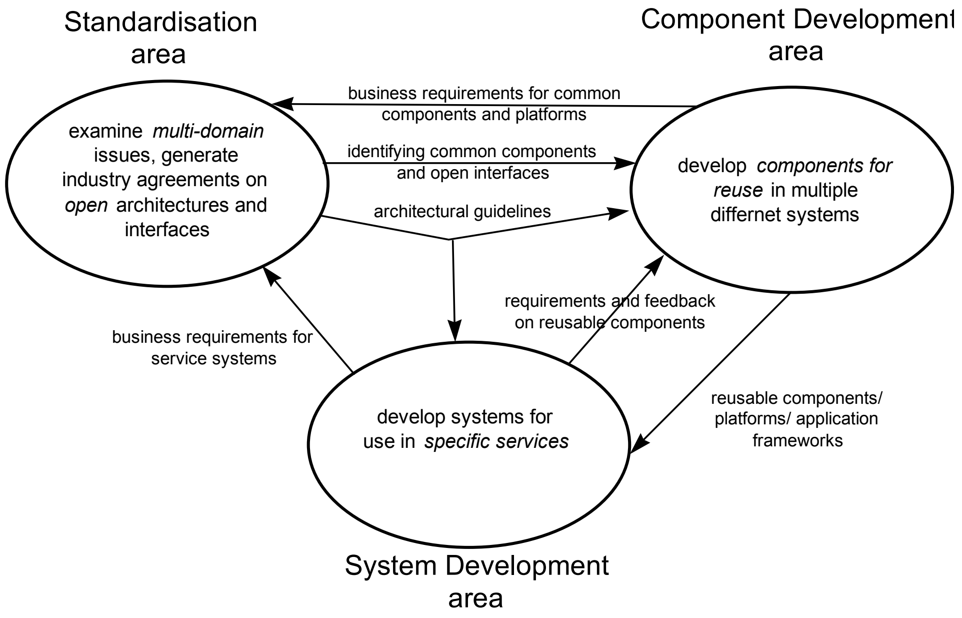
Standardisation area
Component Development area
System Development area
examine *multi-domain* issues, generate industry agreements on *open* architectures and interfaces
develop *components for reuse* in multiple differnet systems
develop systems for use in *specific services*
business requirements for common components and platforms
identifying common components and open interfaces
architectural guidelines
requirements and feedback on reusable components
business requirements for service systems
reusable components/ platforms/ application frameworks

Figure 1: A business model of the telecommunications management software development sector

3. EXISTING APPROACHES

This section provides an overview of the approaches to management system development available from the following industry sectors: telecommunications industry, distributed systems, software engineering and WWW/Java.

3.1 Telecommunications Industry

The development of management system in the telecommunications field has been heavily influenced by the manager-agent paradigm where agents are modeled as collection of managed objects (MOs). The principal examples of this paradigm are OSI Systems Management (OSI-SM) and Internet management. The ITU-T's Telecommunications Management Network (TMN) recommendations [2] build on the manager-agent paradigm by defining an architecture for multi-layer networks of Operation Systems Functions (OSFs) operating in both manager and agent roles that co-operate to manage telecommunications services and networks. The TMN recommendations include an Interface Specification Methodology [3], which provides guidelines for the functional decomposition of management interfaces found at reference points in the TMN functional architecture. This methodology concentrates on generating agent interfaces from management function requirements. It provides no guidance on designing the interactions between the agents and manager functionality of an OSF.

OSI-SM also provides a set of System Management Functions (SMF) that are intended to provide descriptions of commonly required management functions, e.g., event management, log control or test management. SMFs are specified together with the definition of generic attributes, actions and notification for managed objects that perform the function. TMN platforms commonly provide implementations of the SMFs for agent development, often with corresponding manager support. It is also possible to inherit from the SMF MO definitions to provide similar functionality in custom MOs. TMN recommendations contain a similar set of generic, network management related GDMO definitions in [4].

The Network Management Forum (NMF) is a non-profit industrial consortium that aims to provide industry agreements to aid service providers and management system vendors in the procurement and development of TMN-based management systems. The agreements are based on the business needs of NMF members. They are presented as either solution sets, which address specific management problems, or component sets, that represent reusable components that can support solution sets, i.e., aspects of TMN platform functionality such as logging. The solution set provides the solution to a stated business problem and interface conformance statements for resulting products, which together are termed an *ensemble* [5]. An ensemble packages together an outline of what is to be managed, expressed as resources; what functions are required to solve the management problem; and some scenarios to illuminate how these functions should operate in sequence on the resources. The process for developing these ensembles largely revolves around the identification of management functions and MO definitions from existing standards, with new MOs being defined only when absolutely necessary. Packaging information in this format is aimed at making the solution more accessible to industry practitioners and, therefore, more readily reusable than existing formal standards.

Another group addressing the area of telecommunications software development techniques is Study Group 10 of the ITU-T which is addressing languages and general software aspects for telecommunications systems. Major outputs of this group have been Recommendation Z.105 on the Specification and Description Language (SDL), which provides a formal, object-oriented language for describing system behavior, and Recommendation Z.120 on Message Sequence Charts (MSC) which aims to provide a formalized technique for specifying sequence of signals in telecommunication systems. Other areas being addressed by this group are: CHILL (Recommendation Z.200), a high-level programming language for telecommunications systems; testing and verification techniques for telecommunications systems; quality assurance for the telecommunications software process and behavior extensions to GDMO.

3.2 Open Distributed Processing

In response to some of the problems raised by the complexities introduced by large scale distributed systems, ISO has developed the Reference Model for Open Distributed Processing (ODP) [6]. This is a meta-standard intended to guide the development of other standards by providing a framework for the integrated support of distribution, interworking, interoperability and portability, all of which are relevant to service management. ODP approaches the problem of describing distributed systems by expressing the effects of distribution as a number of distribution transparencies, e.g., location, access, or failure transparencies. Distributed systems adhering to the ODP framework are described from five complementary viewpoints addressing enterprise, information, computational, engineering, and technological aspects of a system. These separate viewpoints together aim to provide a complete and consistent view of the system.

Each viewpoint has an associated language for expressing rules relevant to the concerns of that viewpoint. As viewpoints are separate but inter-related views of the same system, the relations between terms in different views are subject to consistency constraints. One aim of ODP-related research is that viewpoint languages could be defined in a formal way that would enable the automation of consistency checks between viewpoints.

The Object Management Group's CORBA standard [7] represents an agreement between a wide range of IT industry players on an approach to integrating distributed object-oriented systems. Though addressing some of the same concerns as ODP, it has concentrated on providing a practical environment for programmers of monolithic systems to easily develop distributed client-server applications. Interfaces to remote objects are defined using an Interface Definition Language (IDL), from which stubs and skeletons can be generated in a variety of languages when writing client and server software. CORBA is proving a viable platform for service management since it provides a better mechanism than existing TMN-based platforms for integrating the wider range of applications found at this level, e.g., accounting databases, customer service desktops, etc. The OMG also specifies a set of CORBA Services [8], which provide commonly required services such as object naming, event propagation, security or transaction. These CORBA Services are defined as object interfaces, expressed in IDL, together with a description of how to obtain the service functionality, including any operation sequencing dependencies. As with SMF, these service definitions can either be provided as implementations in a CORBA product or specialized for use in a specific application.

One body that has already attempted to apply ODP principles and implement the results using CORBA, is the TINA Consortium. This industrial consortium of network operators and telecommunications and computer equipment vendors, aims to develop an open architecture for telecommunications services in a multi-vendor, multi-provider environment. The approach is broad, combining service provisioning principles from Intelligent Networks and telecommunications management principle from TMN, but with ODP providing the overall framework [9]. Though TINA adopted the use of ODP viewpoints, it has not adopted the viewpoint languages suggested in [10] since they were insufficiently precise in their definition. For the information viewpoint, TINA [11] advocates the use of the Object Model Notation defined in [12] (referred to here as the Object Modeling Technique or OMT), which provides a graphical notation for describing objects and their relationships. TINA supplements this with a textual notation supporting quasi-GDMO object definitions and an object relationship model based on the OSI General Relationship Model [13]. This notation provides a representation of objects, including their attributes, the constraints and operations that cause change in the object's state, as well as object inheritance and relationships between objects. For the computational viewpoint TINA [14] again has adopted its own graphical notation consisting of simple component diagrams representing computational objects and the operational and stream interfaces they offer to each other. To provide details of computational object operations and interface structure a superset of IDL, termed Object Definition Language (ODL), is used. ODL allows the definition of multiple interfaces, stream interfaces, references to interfaces used on other objects and the collection of objects into groups termed building blocks. For the engineering viewpoint [15] a distributed processing environment (DPE) is assumed which provides various distribution transparencies required by computational objects (COs) through a set of services made available to engineering objects populating the DPE. The engineering objects themselves are arrived at directly by decomposing the COs. As TINA has adopted CORBA as its DPE, this decomposition consists of mapping ODL to IDL.

TINA goes beyond the ODP by specifying an outline methodology for developing TINA services [16]. These guidelines present a development process where the enterprise viewpoint of a service is addressed during the analysis stage of the development process, the informational and computational viewpoints are both used during the design stages and the engineering model just before the implementation stage. Though the information viewpoint model and computational viewpoint model are described as complementary parts of the design process, exactly how to iterate between them and to describe the relationship between the different objects in these models is not stated in any prescriptive manner. The TINA Consortium has defined service management components using these techniques as part of its Service Architecture [17].

3.3 Software Engineering

The approaches to service management system development described in the previous two sections are very focused on the identification of interfaces between components. This is not surprising since interoperability has always been a major issue in telecommunications and distributed systems and therefore ensuring compatibility of interfaces always features highly in the approaches taken by these industries. The structure of the component behind the interface, often complex compared to the interface itself, has not been exposed to the same push towards common practices within these industries. The software engineering industry however has more of a focus on the whole life-cycle of software, from its analysis and design to its implementation and maintenance. It therefore is a source of useful common practice for development of both a system's interfaces and its internal components. In the case of interface specification, the telecommunications industry has already borrowed many ideas on object-orientation from the software engineering industry, e.g., the use of OMT in TINA.

A very large number of different software development methodologies have emerged from the software engineering community, but with a single "silver bullet" approach proving to be very elusive. One area of convergence, however, has been in graphical representation of object oriented analysis and design entities. This has been promoted jointly by Grady Booch, James Rumbaugh and Ivar Jacobsen, authors of three of the leading methodologies of recent years and now currently working together at Rational Software Corporation. This group observed that though their proposed methodologies were different, often with distinct emphases, many of the graphical notations used were semantically very similar. These were therefore combined into the *Unified Modeling Language* (UML) [18] which provides a single meta model for structuring the components of object-oriented analysis and design models together with guidelines of how they could be visually rendered to users of CASE tools. UML is presented as an open standard, and several CASE tool vendors are supporting it. UML has also been proposed to the OMG as part of its call for proposals towards a common development methodology for distributed systems, thus UML may play an increasingly important part in CORBA-based service management system development. Note however that UML specifically addresses only a notation, and not the process of a methodology.

Of the popular methodologies, the Object Oriented Software Engineering methodology [19] seems worthy of examination in relation to service management because in addition to supporting robust object-oriented structuring of systems, the focus on use cases fits well with the emphasis on end user requirements needed in management systems. Furthermore this technique addresses the whole software life-cycle more comprehensively than some comparable techniques, thus supporting fully the telecommunications service life-cycle models that service management systems must inhabit.

The software engineering community has recently recognized problems with reuse techniques that rely solely on object-orientation. One response to these problems has been to adapt the concept of *Design Patterns* from the architectural and construction industry and apply it to software reuse as demonstrated by Eric Gamma [20]. Design patterns aim to capture the knowledge of experienced designers and document it in a manner that facilitates the communication of architectural knowledge and known design traps to other developers. Typically, therefore, design patterns represent common, well-proven designs. There are several slightly different forms suggested for design pattern documentation, however they all follow a common structure. At a minimum a design pattern states a problem and outlines a solution to it, with a context description that indicates the applicability of the solution. The latter part is key, since the aim is not to sell the pattern to the reader, but to provide the information needed by the reader to enable them to gauge whether the solution presented fits well to the problem with which they are faced and satisfies any other (possibly non-functional) requirements placed upon a possible solution. An important feature of a pattern is a suitable, and preferably brief, name. In this way it is hoped that pattern based terminology will evolve into a more powerful means for communicating between software developers. Great emphasis is placed on expressing patterns concisely and clearly and ideally they should also contain an example of an application or coding of the pattern. Design patterns are typically collected together into *Pattern Catalogues*, though more benefit can be gained for the user when a collection of related patterns, possibly addressing a specific application domain, are carefully cross-indexed, to show how the solutions can work together in different ways. Such an inter-related collection of patterns is referred to as a *Pattern Language*.

Gamma's original pattern catalogue addressed how small groups of software objects solved common problems, however patterns have been written to address a wide range of problems up to and including patterns for structuring enterprises. Mowbray and Malveau [21] suggest that patterns could actually be categorized into levels of architectural scale from those containing a few classes to those spanning several organizations.

3.4 World Wide Web and Java

A recent technological innovation that has had a profound impact on many areas of distributed computing has been the World Wide Web (WWW) and its integration with Java. The ubiquity of WWW browsers makes them an ideal platform for the management work station function, i.e., the application used by management users. The major impact this has on the development cycle of management systems is that it dilutes the need for the definition of open interface between the provider of a management function and a user-operated client. Therefore, instead of having to conform to slowly emerging industry agreements, the provider of a management system can provide whatever management server interface they wish, provided its release is accompanied by that of compatible, WWW-based user applications. This approach is also well suited for user interface development techniques that rely on multiple development interactions based on user feedback, e.g., rapid prototyping. The reusability of Java software is enhanced by the use of JavaBeans, which provides a framework for easily assembling reusable Java components, i.e., Beans, into applications. This is primarily provided through support for a common event, persistency and property customization mechanisms, together with introspection facilities for supporting visual programming.

4. DEVELOPMENT EXPERIENCES

Evaluating the effectiveness of the various approaches to service management system development is difficult since assessments that are made are often not in the public domain, and experience in developing management systems for multi-provider applications is still rare. One source of such information, however, is research projects that have undertaken major management system development and have been in a position to publish their findings on the development process. The results of three such projects are presented here.

The European Union funded ICM (Integrated Communications Management) project has analyzed some of its experiences in developing complex TMN-based systems [22]. Here, the primary objective was the development of a single provider's TMN consisting of several, as yet un-standardized, service and network level management functions. The development process adopted was based on M.3020 [3], however, it was found that to apply this to a collection of management OSFs, the M.3020 process had to be augmented to support the development of distributed management components that needed to interact with each other in both manager and agent roles. This was performed by first decomposing the required management services into management function components. The information flow between components was then analyzed and this was used together with non-functional requirements, e.g., delay constraints, load sharing and information consistency, to determine how the components mapped onto TMN physical building block, i.e., Operations Systems (OSs).

Further TMN system development experience was recorded by the EU-funded project PREPARE [23]. This project undertook the development of several inter-domain service management systems for which few suitable standards were available. The project therefore had to evolve a methodology that enabled it to define new management functions and objects for systems with multiple users and multiple inter-domain management interfaces [24]. The process developed centered on the use of scenarios to define the often complex interactions between multiple management users and systems. The initial requirements' analysis, termed enterprise modeling, was performed using a technique developed in the ORDIT project [25]. This involved an analysis of the enterprise context of the various systems being developed. This analysis was stated in terms of the organizations that were stakeholders in the service being managed and the human actors operating within those stakeholders, e.g., account managers, network managers and service end users. The requirements were then extracted by stating responsibilities between actors and the obligations they fulfilled to discharge those responsibilities. These obligations were then refined to be expressed in terms of a set of activities operating on a set of resources. To show the relationships between the different activities performed by different actors, inter-domain management scenarios were drawn up for the interactions that involved more than one stakeholder. For example one scenario was that a fault in an ATM bearer service provider is propagated up a value chain of providers resulting in a user being informed of the fault's effect on a multimedia conferencing service. The model of stakeholders, actors and their relationships was then used as the basis for defining a TMN functional architecture containing network element functions (NEFs), network and service level OSFs and work station functions (WSFs) and for identifying the reference points needed between them. In this respect, the approach followed emphasized the definition of inter-domain interactions rather than the intra-domain interactions that were the focus of the ICM approach. The activity and resource descriptions were then used to develop the interfaces that would instantiate these reference points as GDMO MIB definitions. The scenarios at this stage were refined into sequence diagrams showing CMIP interactions between different NEFs, OSFs and WSFs. This approach resulted in the successful development of complex prototypes of an open service market management situation involving several OSFs, and WSFs.

Though such a research project does not reflect completely many of the pressures on the development process present in a commercial environment, it still enables us to assess some of problems involved. Principally these were encountered in the large leap that needed to be taken by designers in developing interface MIB definitions from the activities, resource and scenario descriptions. The refinement of scenarios provided a good medium for ensuring that the interfaces were consistent with the original requirements. However working from this basically functional requirements' analysis put the burden of reaping the benefits of object-orientated reuse in GDMO definitions largely on the intuition and experience of the designers. In addition, in defining GDMO interfaces which supported complex inter-domain interactions, it was found very difficult to express the behavior of individual MOs in a way that made the behavior of an OSF with multiple interfaces clear to its implementers. The refinement of scenarios into CMIP interaction diagrams helped somewhat in clarifying the OSF's external behavior, however a clear route to designing the OSF's internal design structure was still lacking. An additional advantage of the use of CMIP interaction diagrams, however, was that they readily lent themselves to forming the basis of test specifications for the final systems.

Direct experience of applying the ODP viewpoints to the area of multi-domain management systems has been obtained in the EC funded project Prospect [26]. This experience was gained in the context of the development of a tele-education service which delivered its course content to users through an integration of; multiple multimedia tele-services; a Virtual Private Network (VPN) service and an ATM bearer service, all obtained from separate service provider organizations. The challenge, therefore, was to provide management services for this composite tele-education service, through the integration of the management services operated by these various subcontracting organizations. The management functional areas addressed were configuration, subscription and accounting management. ODP was selected as an approach here since it allowed the design of the systems developed to be expressed in a manner easily comparable with other contemporary research, development and standardization initiatives that are also working towards aligning with ODP, e.g. TINA-C and ITU-T's working group on Open Distributed Management [27].

As discussed previously however, ODP does not provide a development methodology, so one had to be adopted that suited the needs of the project while being faithful to the use of ODP viewpoints. The method followed was influenced both by the experiences in the similar problem domain of the PREPARE project and the use of ODP within TINA. The method emulated PREPARE in using ORDIT for the initial requirements' analysis before identifying management functional components at a very high level and describing their interactions through a set of scenarios in order to refine the requirements. The selection of high level functional components was driven largely by the reuse of existing designs for service management, taken in some cases from PREPARE (e.g. multimedia conferencing, VPN and ATM bearer configuration management), as well as more general service management components from TINA 1994 Service Architecture (i.e. subscription and accounting management). This high level functional decomposition was expressed as a computational model, with CO interfaces identified where required to support information flows.

Using the scenarios to retain a focus on the user requirements, the information and computational models were developed for the various components. For components being designed from scratch or being modified from existing designs, only outline information models were developed using the OMT notation, detailed models using GDMO or quasi GDMO were not attempted. The computational models were found however to be where most of the detailed design effort was centered, primarily since this led most directly to the engineering model and hence implementation. Here, the refinement of the scenarios as interaction diagrams between COs was found to be a very useful development tool. Such diagrams helped in the identification of the operations needed at the interfaces between COs, this being especially important where the interface lay between COs being developed by different project partners. The interaction diagrams were also found useful as a method for clarifying and validating the relationship between Information Objects (IOs) and COs, if only in a somewhat informal manner. When incorporating components from TINA which were documented as separate information and computational models, the generation of additional sequence diagrams was necessary to clarify the relationships between IOs and COs sufficiently to support extending these designs and to provide the detail needed for implementation.

As many of the components were to be implemented using a CORBA platform, the detailed interface descriptions of the COs were specified in IDL. Separate CO server interfaces were defined as individual IDL interfaces grouped in an IDL module definition representing the CO. Where components where based on CMIP platforms the interfaces were defined in GDMO, these being converted where necessary to IDL for use in CORBA-CMIP gateways, using compilers based on the X/Open's JIDM recommendation [28].

5. DISCUSSION

These experiences show that interface-centric development approaches, such as that in M.3020, are, by themselves, inadequate for supporting the development cycle of the service management systems. The need for such a system to perform interactions over multiple interfaces requires a development approach that addresses both the system's external interfaces and their relationships with the system's internal state and operation. ODP provides one approach for relating these different concerns, through the use of viewpoints. The computational viewpoint supports functional decomposition and interface definition, and the information viewpoint supports object-oriented information modeling, so both support essential parts of the development process. However, the experiences in Prospect indicate that the difficulty in moving smoothly between these viewpoints in a design process reduces the effectiveness of addressing these different concerns in this way.

Similar arguments can also be applied to reusable components. Though, ideally commercial reusable component should consist of some required function visible only through a public interface, the complex nature of service management systems, and their rapidly changing requirements may limit this approach. If components are to be used at different levels of architectural scale, then more complex components offering multiple, interrelated interfaces must be presented with some exposure of their internal operations and the dynamic relationships between external interfaces and internal state. This is even more essential if a component is to be adaptable to changing requirements. Such an approach is being examined in the latter stages of the Prospect project, where both systems and reusable components are being developed and documented through a combination of use case descriptions, object models combining internal information and external interfaces, and sequence diagrams mapping the use cases onto the object model [29]. This approach is analogous to that used for NMF ensembles, where the requirements (use cases), solution (object model) and usage scenarios (sequence diagrams) are packaged together to make industry agreements to management problems more readily accessible. A further analogy can also be made to design patterns, were a similar problem-solution-context form is used. Currently, design patterns are used to publicize proven solutions commonly used by experienced developers. A novel application of the design pattern form, however, could be in the publication of standardized components. As the number of standardized components increases, this could greatly ease the task of identifying candidate solutions for a particular application. In addition, the use of the pattern form in the development of standardized components may accelerate their acceptance, by revealing the key aims behind interface conformance specifications. The concept of pattern languages could also be used in the structuring of families of standards, by emphasizing the relationships between specific components. This approach has already been adopted in some text books covering COS, including [21]. It is not expected, however, that patterns could replace standards, since the latter are still needed to provide consistent and unambiguous definitions of interfaces. Instead, however, patterns may make the aim and application of standards more accessible than their current, necessarily complex and detailed form. Thus, users of standards would work initially with patterns referring to points of conformance, rather than descriptions appended to conformance statements, as is the case with many existing standard formats.

6. CONCLUSION

Service Management systems are under pressure to respond to the rapidly changing requirements of the open services market. A methodology for their development must accommodate the generation of and conformance to open interfaces, as well as adopting reuse techniques to support flexible and robust designs. Existing interface-oriented methodological approaches are found inadequate without augmentation to link the interface design with internal data models of systems and the dynamic relationships between the two. ODP viewpoints provide one approach that may satisfy these requirements, but experience in their application found the viewpoint languages available and the inter-viewpoint mapping insufficiently mature for this application domain. An alternative approach using UML representations of use cases, object models and sequence diagrams is seen as more promising and is currently under evaluation. The similarity between this latter approach, NMF Ensembles and design patterns reveals the potential benefits of embracing the design pattern form as the principle mechanism for developing and presenting standardized components.

7. ACKNOWLEDGEMENTS

This work was partially sponsored by the European Commission under both the PREPARE project (contract R2004) and the Prospect project (contract AC052). The views stated here are not necessarily those of the Prospect consortium.

8. REFERENCES

[1] E. Adams and K. Willetts, *The Lean Communications Provider: Surviving the Shakeout through Service Management Excellence*, McGraw-Hill, 1996, ISBN 0-07-070306-X.

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

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

[4] *Generic Network Information Model*, ITU-T Recommendation M.3100, 1992

[5] *The "Ensemble" Concepts and Format* NMF025, Issue 1.0, Network Management Forum, Morristown, 1992.

[6] Information Technology- Open Distributed Processing- Reference Model- part 1: Overview, ITU-T Draft Recommendation X.901/ ISO/IEC Draft International Standard 10746-1, 1995.

[7] *The Common Object Request Broker Architecture and Specification*. OMG Document Number 92.12.1, Rev. 1.1, Object Management Group, 1992.

[8] *Common Object Services*, Volume 1 and 2, Object Management Group, 1995.

[9] P. Prozeller, *TINA and the Software Infrastructure of the Telecom Network of the Future*, Journal of Network and Systems Management, Vol. 5, No 4., December 1997.

[10] Information Technology- Open Distributed Processing- Reference Model- Part 4: Architectural Semantics, ITU-T Draft Recommendation X.904/ ISO Draft International Standard 10746-4, 1995.

[11] H. Christensen and E. Colban, *Information Modeling Concepts*, TINA Baseline Document, TB_EAC.001_1.2_94, April 1994.

[12] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, N.J., 1991.

[13] Information Technology- Open Systems Interconnection- Structure of Management Information- Part 7: General Relationship Model, ITU-T Recommendation X.725/ ISO/IEC Draft International Standard 10165-7, 1994.

[14] N. Natarajan, F. Dupuy, N. Singer, and H. Christensen, *Computational Modeling Concepts*, TINA Baseline Document, TB_A2.HC.012_1.2_94, February 1994.

[15] P. Graubmann and N. Mercouroff, *Engineering Modeling Concepts (DPE Architecture),* TINA Baseline Document, TB_NS.005_2.0_94, December 1994.

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

[17] H. Berndt and R. Minerva, *Service Architecture*, TINA Baseline Document TB_MDC.012_2.0_94, March 1995.

[18] H. Eriksson and M. Penker, *UML Toolket*, Wiley Computing Publishing, ISBN 0 471 19161 2, 1998.

[19] I. Jacobsen, M. Christerson, P. Jonsson, and G. Overgaard, *Object-Oriented Software Engineering*, Addison-Wesley, 1992, ISBN 0-201-54435-0.

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

[21] T. Mowbray and R. Malveau, *CORBA Design Patterns*, Wiley, 1997 ISBN 0-471-15882-8.

[22] D. Griffin (Ed), *Integrated Communications Management of Broadband Networks*, Crete University Press,1997, ISBN 960 524 006 8.

[23] J. Hall (Ed), *Management of Telecommunications Systems and Services: Modeling and Implementing TMN-Based Multi-Domain Management*, No. 1116 Lecture Notes in Computer Science, Springer-Verlag, 1996, ISBN 3-540-61578-4

[24] D. Lewis, T. Tiropanis, L.H. Bjerring, and J. Hall*, Experiences in Multi-domain Management Service Development*, Proceedings of the 3rd International

Conference of Intelligence in Broadband Services and Networks, Heraklion, Greece, October 1995, ISBN 3-540-60479-0.

[25] R. Strens and J. Dobson, *Responsibility Modeling as a Technique for Organizational Requirements Definition*, Intelligent Systems Engineering, vol. 3, no. 1, pp.20-26, 1994.

[26] V. Wade, D. Lewis, M. Sheppard, M. Tschichholz and J. Hall, *A Methodology for Developing Integrated Multi-domain Service Management Systems*, Proc. of 4th International Conference on Intelligence in Services and Networks, Como, Italy, May 1997, Springer-Verlag, 1997.

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

[28] Inter-Domain Management Specifications: Specification Translation, X/Open Preliminary Specification, X/Open, Reading, Draft of April 17, 1995.

[29] V. Wade, D. Lewis, W. Donnelly, D. Ranc, and N. Karatzas, *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.

9. BIOGRAPHY

Dave Lewis graduated in Electronic Engineering at the University of Southampton in 1987 and received an MSc. in Computer Science from University College London in 1990, where he has worked since, as a research fellow in the Computer Science Department. He has worked primarily on the EU funded projects PREPARE and Prospect, in which he has been responsible for planning high speed international testbed networks and for leading teams developing integrated, multi-domain service management systems. He is also working on a Ph.D., researching a service management development architecture for the open services market.