A Service Engineering Approach to Inter-domain TMN System Development

David Lewis, Thanassis Tiropanis, Rong Shi
Computer Science Department, University College London
Gower Street, London WC1E 6BT, United Kingdom
Tel: +44 171 391 1327, +44 171 419 3687, +44 171 419 3249
Fax: +44 171 387 1397
E-mail: D.Lewis@cs.ucl.ac.uk, T.Tiropanis@cs.ucl.ac.uk, R.Shi@cs.ucl.ac.uk

Alexander Richter
GMD-Fokus
Hardenbergplatz 2, D-10623 Berlin, Germany
Tel: +49 30 25499 287, Fax: +49 30 25499 202, E-mail: richter@fokus.gmd.de

1. Introduction

In recent years most work in the area of distributed and telecommunication-based systems has been under-pinned by an assumption that the resulting standards, interfaces and products will have to operate in a global open market for telecommunication services and software. Such a market will be characterised by a much more crowded and diverse arena of players than has been apparent in the traditional telecommunications industry structured around large state-owned enterprises. The increased competitiveness of this global market will produce ever increasing pressures to reduce costs and to increase service quality to the customer. Open interfaces have an important part to play here both in reducing the burden on the provider of dealing with multiple proprietary interfaces and increasing choice to the customer by encouraging competition on the quality and cost of service rather than ties to specific interface technologies.

Another area seen as being of primary importance to cost reduction and customer satisfaction is that of management. This covers the management of the providers resources to obtain optimum efficiency, the management of the delivery of the service to the customer and the integration of these activities to maximise the effectiveness of both. Open interfaces have long been a major feature of network resource management, primarily through the protocols and standards developed by the IETF (SNMP), ISO (OSI-SM) and ITU (TMN). These schemes were, however, conceived to address the needs of large corporate data communication networks and monolithic public telecommunications networks. The additional requirements imposed by the open services market have been addressed to an extent by the ITU-T in the TMN recommendations with the separation of service and network related functions, and the identification of inter-domain interfaces. Though this area has been the focus of much recent work, most notably by the Network Management Forum, the TINA Consortium and several EU funded RACE projects, the breadth of practical experience in this area is still relatively limited.

This paper presents some recent work in the practical application of open standards to the area of service management. This work, undertaken in the RACE II project PREPARE, was conducted in an environment specifically designed to impose many of the requirements of an open service market on the development of a system of interworking service and network management components; in this case managing multimedia teleservices over a broadband network. The next section outlines some of the standards that influenced this work. Section 3 describes the specific context in which this prototyping work was conducted. Section 4 details the design approach taken, the components used and the experiences gained before some conclusions are drawn in section 5.

2. Current Architectures

The most mature open architecture applicable to service management, and the basis chosen for the PREPARE project, is the ITU-T M.3000 series of recommendations on the Telecommunications Management Network (TMN) [M.3000]. This standard has been widely accepted by the telecommunications industry in Europe and now increasingly in North America and Japan. The core architecture recommendation [M.3010] structures management functionality into a set of levels starting at the network element level, through the network level to the service level and finally the business level. TMN compliant standardisation efforts have led to a wide range of interfaces being developed at the network element level and more recently at the network level. Open interfaces at the service level

are only now starting to emerge, primarily intended for operation at the inter-domain interface identified in TMN, i.e. the X interface. The Network Management Forum (NMF) is particularly active in this area through its SMART group. This group is addressing specific areas of management functionality identified in a common Business Process Model [NMF-BPM], such as trouble ticketing, service ordering and billing. The results of this work are Ensemble documents [NMF-025] that profile the use of existing standards and, if necessary, provide new information models, to address the specific problem under consideration. What is not prescribed is how the resulting management components might be integrated with each other or with the management components related to network and network element management problems developed in the NMF's parallel OMNIPoint initiative [NMF-SF].

Comment: I think the sentence needs some improvements, because ... to address the specific problem being addressed... sounds not very good. Unfortunately, I have no idea how to rephrase it.

The process of defining open interfaces in TMN concentrates on the development of information models defined in terms of managed objects (MO) that can be accessed using the management information retrieval protocols SNMP and CMIP. The functionality of a TMN interface is not formally defined in an information model but may be described informally through MO behaviour descriptions as well as through the relationships between a group of MOs. NMF Ensembles complement these static information models with information flows showing how an interface may be used in a dynamic sense, as well as conformance statements. The TMN Interface Specification Methodology [M.3020] provides guidelines for defining interfaces in terms of management interfaces that are decomposed into Management Service Components (MSCs) which are in turn decomposed into Management Functional Components (MFCs). However this methodology is primarily intended for the design of interfaces, and in practice MSCs and MFCs are not often utilised as units of reusability for interface specifications, this being hampered further by a lack of a strict notation for these components. MOs are frequently reused between specifications, however without a clear mechanism for reusing MOs in groups forming MFCs the real value of this reuse tends to be more in the interface syntax of an MO than in the functional semantics it may offer. A further large factor restricting widespread reuse of functional component in TMN is the current lack of a common API for management platforms; this is currently being address by the X-Open and the NMF [SPIRIT]. This lack of clear functional reusability in TMN presents a problem for those engineering service management systems for the open service market, where reusability will play a vital part in responding quickly to ever changing requirements on service. This is further compounded by the functional decomposition of managing applications being outside the scope of TMN interface specifications.

A different approach to component reusability has been taken by the Open Management Group (OMG) in developing its Common Object Request Broker Architecture (CORBA) [CORBA]. Drawing its membership from the wider data processing industry, as opposed to the telecommunication industry sector that influenced the development of TMN and the NMF, the OMG aims to apply to the area of distributed processing the object oriented techniques common in software engineering today, e.g. use of C++ classes as units of functional reuse. This is performed by providing mechanisms to support distribution transparencies, that have been identified as important in separating the design of a distributed system from the mechanisms providing, and the problems presented by, the distribution itself.

The OMG approach is in tune with ISO Open Distributed Processing recommendations [X.901] that define a set of such transparencies covering access, location, transaction, fault etc. Significantly, the ODP recommendations go beyond outlining a technique for arriving at an interface specification and propose a general approach to defining distributed systems. This is based on the five ODP viewpoints: the enterprise viewpoint aiming largely at requirements capture; the technology viewpoint for defining the technological constraints placed on a system; the information viewpoint addressing the information to be processed in the system; the computational viewpoint addressing how the system is decomposed into computational objects and the engineering viewpoint describing how information and computational models are implemented as engineering objects, e.g. C++ classes.

Though the adoption of ODP techniques encourages the development of a clear, object-oriented information model the availability of object oriented platforms providing the appropriate distribution transparencies, e.g. a CORBA implementation, leads to the use of the computational objects of the computational viewpoint as the primary units of system decomposition and eventual reuse in the engineering model. The reusability of such computational objects, and also their implementation in corresponding engineering objects, is aided by the ODP concept of clustering which allows for the grouping of engineering objects into larger units of functionality. This provides system designers with a wider and more continuous range of granularity of functional reuse than seems to be practiced in TMN.

Though the ODP standards and the OMG architecture are applicable to any distributed processing problem, the Telecommunications Information Networking Architecture Consortium (TINA-C) have attempted to apply these principles to the needs of the telecommunications industry [TINA-018]. This has involved the adoption of ODP as the basis for their architectural approach resulting in the definition of an abstract Distributed Processing

Environment (DPE) [TINA-005]. To provide continuity with existing telecommunications industry standards TMN principles have been adopted for the management aspects of this architecture while the influence of the IN standards [Q.1204] is seen in the approach to functional decomposition and reuse. By bridging the gap between the emerging distributed processing industry and the well established telecommunications industry in this way, TINA provides a path for TMN based management architectures to be integrated with the ODP-based techniques better suited to the requirements of the open service market. The following sections give an example of how this path could be followed based on some of the practical experiences gained from the PREPARE project.

3. Open Service Market Context in PREPARE

As stated in the previous section, the PREPARE project offered an opportunity to investigate the development of management services in an open service market context and in an environment that imposed requirements down to the implementation level. This consisted of an enterprise model typical an open service market situation and requirements to manage broadband networks and multimedia teleservices [Bjerring].

Comment: Is there still a reference missing ???

The enterprise model chosen consisted of a value chain of service providers. At the lowest level multiple public network operators (PuNOs) provide an ATM Virtual Path service over their respective network domains, and conspire to provide this service, spanning several domains, via a single management. This management interface is used by a Virtual Private Network (VPN) provider, in concert with the management of private network resources of ATM LANs, owned by private network operators (PrNOs), to provide end to end management services connecting hosts on the ATM LANs. This VPN service is in turn used by providers of multimedia teleservices, namely a multimedia conference (MMC) service and a multimedia mail (MMM) service, in the provision of these teleservices to a customer organisation between internationally distributed ATM LAN sites.

The decomposition of management functionality between these various organisational stakeholder was performed initially at a very low level of granularity and in line with the TMN functional architecture. This involved identifying one TMN Operations System Function (OSF) for each stakeholder at the service level with additional OSFs, if required, at the network level and network element level via Q-adapter functions. The aim was to implement this TMN architecture as a set of inter-operating Operations Systems (OSs) mapped simply to the OSFs. *Figure 1* shows schematically how these OSs were split between organisational stakeholders and mapped over a broadband demonstration network, spanning London, Berlin and Copenhagen, and consisting of ATM LANs and ATM cross-connects (XCs). This TMN architecture provided an good testbed for examining the requirements on service management services and their constituent components as described in the following sections.

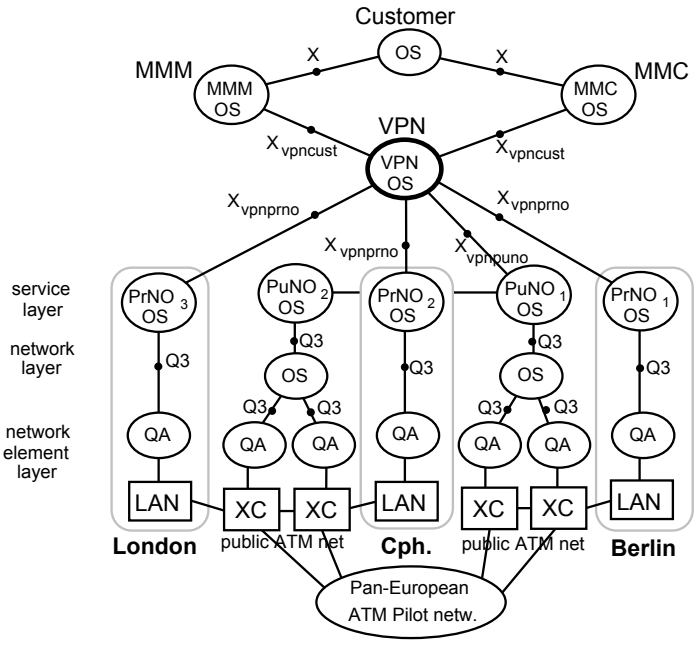

Figure 1: Overall TMN Physical Architecture for PREPARE

4. Applying Service Engineering Principles to TMN OSF Design

The enterprise context for the VPN, the MMM and the MMC services in PREPARE generated several requirements which had to be mapped to the definition of the information and computational models for the management of each service. The complexity of the enterprise model made the design even more challenging. The MMC and MMM service providers were to be customers to the VPN service provider, which in turn would be a customer to the ATM Virtual Path (ATM VP) service provider. The design of the services had to be open enough to support a multi-player environment, with customers with diverse needs. The development and integration of the services had to be shared between different companies which put an extra requirement for well defined interfaces between modules that could be developed separately.

The decomposition of the management logic for the services into reusable components that communicate with each other by using well defined interfaces and the use of object orientation proved to satisfy the above requirements to a good extent.

4.1 Decomposing services into service components

In TMN terminology, the software modules that implement management logic are defined as operation systems (OS). OSs for each administrative domain were derived from the enterprise specification in PREPARE. For the management of services, several service operation systems (S_OS) were defined. The communication between S_OSs in different domains was accomplished over the TMN X interface. A common information model in GDMO and ASN.1 specified the information to be exchanged over the X interface. A set of event-trace diagrams defined the X interface interactions between the S_OSs for a set of different service management scenarios.

In a multi-player environment the same physical entity (an organisation or an individual) can assume many different roles. For each of them, the appropriate service management logic is required. For example, an MMC service provider that uses VPN resources to provide its customers with end-to-end connectivity has two roles: the one of the MMC service provider and the one of the VPN service customer. A set of application modules is required for each of these roles. If these modules can be identified, they could form reusable entities of service or management logic. In TMN terms these reusable management modules are defined as Management Service Components (MSCs).

Comment: This is the first and the last time that we are talking about an application module.

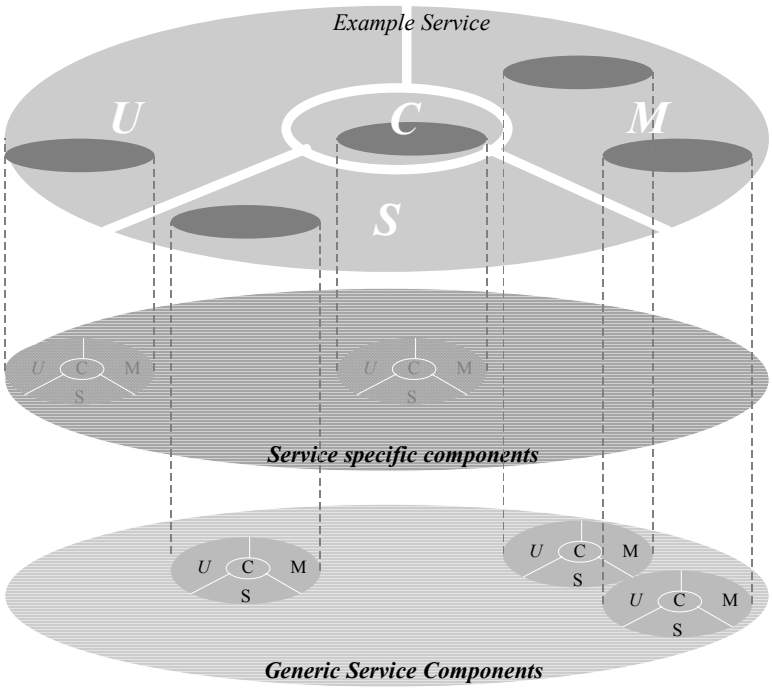

Figure 2: The structure of services and service components

Comment: This figure prints out rather poorly at least on one of our printers. In case this figure will be photocopied it will get even worse.

In PREPARE, TINA-C guidelines were followed for the decomposition of services and management logic to Service Components.

The TINA-C architecture provides for Service Components (SCs) that can be reused between services and which are categorised to the following groups:

- *Usage:* components that provide interfaces to the customers or users of a service
- *Management:* components that provide management functionality to services
- *Substance:* components that provide access to other services or resources
- *Core:* components related to the core service functionality, independent of management, usage or use of other services.

Comment: I have reformatted this paragraph

Therefore, the service logic of a service can be decomposed to several SCs that belong exclusively to one of the above areas. The TMN MSCs can be seen as TINA SCs that belong to the management group of SCs. This structuring guideline, known as the Universal Service Component Model (USCM), can be applied recursively to the internal structure of the SCs as shown on *Figure 2*.

The SCs identified in PREPARE were mostly related to the management area of services. They were further categorised to *generic SCs* and *service specific SCs*. The generic SCs are the reusable modules that seem to be common for most services and they mainly facilitated generic service management functionality. The service specific SCs dealt with requirements for service specific functionality in the areas of management, usage, substance and core. *Figure 2* shows the structure of the services and the service components according to the USCM and the separation between generic and service specific SCs.

The USCM was applied to the computational and engineering modelling of services in PREPARE. The service components were mapped to computational objects directly. The usage, management, substance and core interfaces to each separate service component were mapped onto separate engineering objects.

The following list includes the *management SCs* that were *common* for all the services.

- The *service offerings browser SC* (SOB) offers functionality for browsing through adverts for services on offer. The adverts are placed as entries on the X.500 directory. A special information model was developed to model the information provided in service adverts. This service component can exist as a separate service by itself. In the context of service management though, it offers pre-service management functionality and therefore it consists of a generic service component that can be part of any service.
- The *subscription manager SC* (SUM) implements all the functionality needed for managing existing subscriptions to services. The SUM provides with means for initialising subscription information and locating subscription specific management information in other domains (including those not under the direct control of the service provider). This was performed by employing a unique identifier for each service subscription and was also used for demultiplexing subscription specific service requests exchanged between SCs.
- The *service information locator SC* (SIL) provides functions for locating service management agents on other customer or provider domains. This feature can be used by other components that invoke operations over the TMN X interface. For a given subscription, this SC can return the address of the service management agent on the provider or customer domain.
- The *bill manager SC* (BM) offers the utilities required for collecting, browsing, and paying of bills between providers and customers for each specific subscription.
- The *Graphical User Interface co-ordinator SC* (GUI_C) implements the functions for integrating the GUIs of the several *service components* installed and running on the same application. It makes the addition of new SCs more convenient by integrating their graphical user interfaces with the existing ones behind a common front-end.

The relationships between the computational objects on which each of the above service components can be mapped is shown in *Figure 3*.

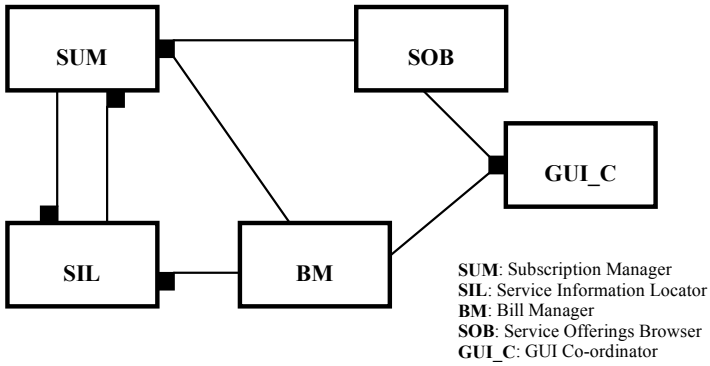

Figure 3: Relationships diagram for the common Service Components

Comment: There is no CM! It has to be the SUM!!

For each of the services, the common SCs were used together with *service specific* SCs. Service specific SCs were modelled for the MMC, MMM and VPN services. As an example of service specific SCs, components on making up the VPN customer functionality consisted of one to manage VPN end points, one to manage the reservation of resources between these end points, one to manage the allocation of resources between the end points, one for monitoring the status (fault, availability etc.) of the service and one that provided a graphical user interface to this functionality.

4.2 Building-blocks and contracts

Service components can be grouped together according to the dependencies between them. Grouping of service components allows for their better management. They can be treated as an entity that can be moved or copied to other locations and plug into other applications by offering a set of interfaces to them. The notion of a *building block*

(BB) in TINA-C is that of a group of computational objects that has these properties. The term *contract* is used for the interfaces to the building blocks.

Comment: What exactly is a contract? This description is not very detailed.

The relationships between the SUM and the SIL in *Figure 3* indicate the strong dependencies between them. For this reason they were put together in the *subscription BB*. The other SCs were organised into separate BBs. The *bill BB* includes the BM, the *service offering browser BB* includes the SOB and the *GUI co-ordinator BB* includes the GUI_C.

Most of the above BBs include an extra *co-ordinator SC* that is responsible for the initialisation, termination, activation and deactivation of the *contracts* of the BB. This component is also responsible for other generic building block management functions.

The *service specific components* were organised into separate BBs. There are two BBs for each service: the customer BB and the provider BB. For the VPN service, there are the *VPN service customer BB* and the *VPN service provider BB*. The same applies for the MMC and MMM services.

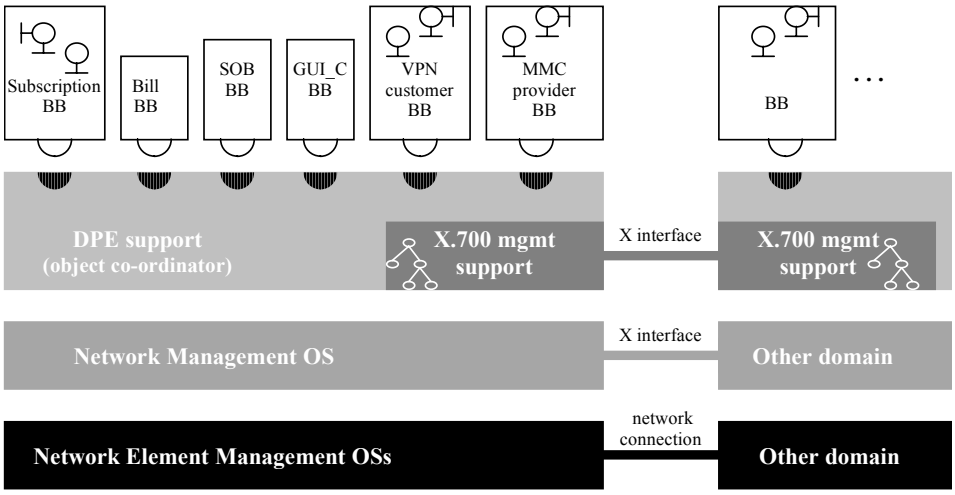

Figure 4: Structure of TMN S_OSs with building blocks and contracts

Comment: The 'Bill BB' should read 'BM BB' and at least an arrow has to point to the object co-ordinator SC. I think it is the half circle below each BB.

A picture of all the BBs organised on the site of the MMC provider is given in *Figure 4*. This combination of BBs comprises the VPN customer OS and the MMC provider OS and shows the case of being MMC provider and VPN customer at the same time. The DPE support and the X.700 management support entities are explained in detail in 4.3.1 and 4.3.2 respectively.

4.3 Key service components

Since no CORBA or TINA compatible platforms were used in PREPARE, two special service components were developed. The *object co-ordinator* was to support TINA DPE functionality. The *X.700 management object* was to support an X.700 management gateway to the DPE. It was used as a special DPE service by other service components that utilised the TMN X interface. The TMN X interface was used only for inter-domain transactions in PREPARE as shown in *Figure 4*.

4.3.1 The object co-ordinator

Although in *Figure 4* the object co-ordinator appears to be one layer underneath the other service components, it can also be considered as a special service component. It provides location transparency to all the contracts on a global level, by means of a unique global name it assigns to each of them.

The object co-ordinator offers a contract, with a well known reference, for the other contracts to use. It consists of the following methods:

- *Register*. This method is invoked by a contract of a BB when it registers with the object co-ordinator for the first time. A desired name can be passed as an optional parameter. If it is unique, the object co-ordinator will accept it, otherwise it will come up with an alternative unique name for it. Other parameters include the location of the calling object.
- *Deregister*. This method is invoked by a contract when it wishes to deregister.
- *Reregister*. This method is invoked by a contract when it wishes to change its registration context (except for the unique name). The most common case is when the interface changes location and wishes to let the object co-ordinator know of its new location. This can provide *object mobility support* that can be used for *personal* or *terminal mobility*.
- *Call*. This method is invoked by a contract when it wishes to call a method provided by another contract. The call is made via the object co-ordinator which knows the location of all the contracts. If the call is successful, the object co-ordinator will return the results to the calling object. If it is not, it will forward the returned error message to the caller.

For scaleability reasons, many object co-ordinators can be instantiated across the network. They can set up connections between them and via special interfaces they can:

- *Locate* interface objects registered with any other object co-ordinator.
- *Check the uniqueness* of a global name when an object registers.
- *Invoke* methods on interface objects registered with any of them.

On a high level, an object co-ordinator resembles a DPE or an Object Request Broker (ORB), while the communication between different object co-ordinators resembles the concept of the Kernel Transport Network (KTN) in TINA. In PREPARE, the object co-ordinator was used only for intra-domain access. All the transactions between different domains were made using the TMN X interface. Therefore, the inter-co-ordinator access was restricted to the boundaries of each administrative domain.

Figure 5 shows the operation of the object co-ordinator during an invocation by one of the interface objects (A) of a method of another interface object (K) registered with a remote co-ordinator. In this case, object co-ordinator 1 forwards the call to all the other object co-ordinators. But only object co-ordinator 2 (with which the called interface object is registered) replies by making the call to the interface object and returning the results to object co-ordinator 1, which in turn returns the results to the caller (A).

The object co-ordinator proved to be a core component for applying and trying out the TINA-C principles on our design. Although we cannot claim that it replaced all the functionality a DPE can provide, it responded to our requirements for a DPE-like component to a great extend. Its simple design made changes and optimisations flexible and gave us a chance to look into problems concerning the use of distributed platforms.

4.3.2 The X.700 management object

The X.700 management object is a component that was developed as a DPE service. It offers an X.700 management API to other components and can handle X.700 *operations*, *actions* and *event reports*. In PREPARE, it was used by contracts that utilised the TMN X interface for inter-domain management transactions.

A special implementation of this component also offers access to the X.500 directory. The combination of the X.500 directory and X.700 management provides location transparency for X.700 agents that have an entry in the X.500 directory [CFS-H430].This entry also consists of address information to contact an X.700 agent. Due to similar naming conventions for X.500 data objects (directory objects) and X.700 data objects (managed objects) these names are combined by using special rules and creating a unique name for each X.700 data object regardless which agent is responsible for it.

This component can exist independent of DPE support. When installed as a DPE service it provides an *X.700 gateway* facility to DPE-registered interfaces.

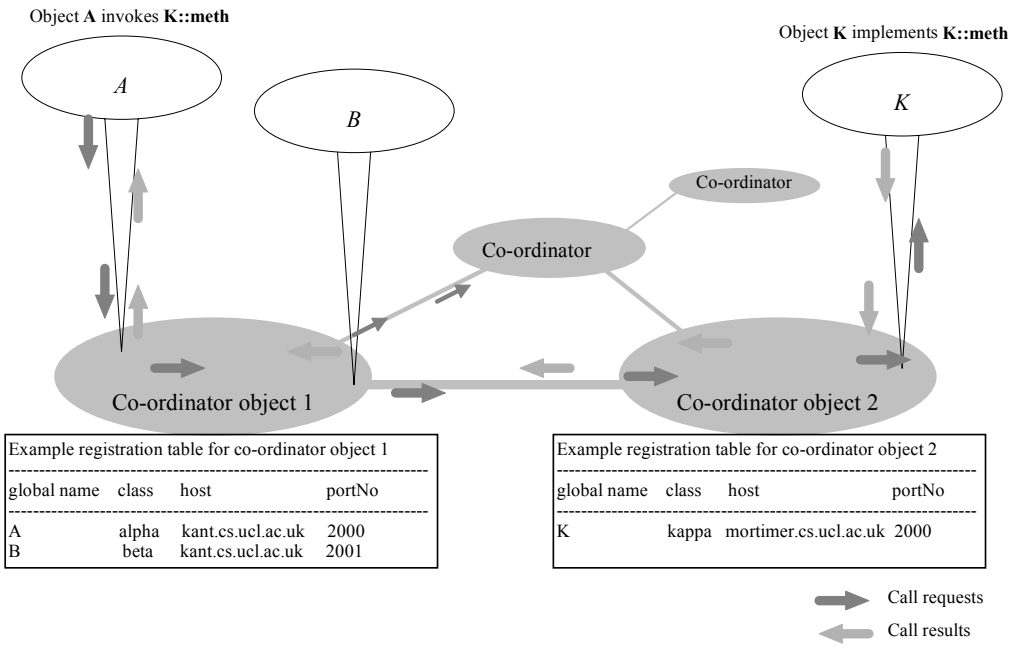

Figure 5: Utilising interfaces via the Tcl Object Co-ordinator

Comment: This figure belongs to section 5.3.1

4.4 Technology used

The object co-ordinator and all the service components were implemented in *Tcl* which satisfied requirements for *flexibility* and *simplicity* when developing service management logic.

Tcl is an *interpreted scripting language*. As a high level language it can be used by service managers who are not necessarily experienced programmers. Since it is *embeddable* into C or C++ code, it is also possible to reuse already existing software and integrate it with Tcl code.

The *Tcl Object Co-ordinator* (TOC) is the implementation of the object co-ordinator component. It is based on Tcl (version 7.4) [TCL-TK], the object-oriented Tcl extension ObjectTcl (version b1.1) [OTcl], and the distributed programming Tcl extension Tcl-DP (version 3.3b1) [Tcl-DP].

There are two implementations of the X.700 management object. One is based on *tcl-rmib* which is part of the *OSIMIS 4.0* management platform [OSIMIS]. The other is based on *tcl-idmis* which is part of the *ANDROMEDA* [Dittrich-95d] platform, which, in turn, is based on OSIMIS and ISODE.

Both, tcl-rmib and tcl-idmis, offer a Tcl-based X.700 management interface, where management operations are invoked by Tcl commands. The parameters passed to and the results returned from the X.700 management object are always strings. To support the asynchronous reception of event reports they are mapped to Tcl call-backs. Tcl-idmis integrates X.500 and X.700 under the same API and therefore offers location transparency for X.700 management agents.

5. Conclusion

Though the PREPARE open service market context is a fairly limited example of potential enterprise situations that could evolve it gives us some insight into the problems that a management system designer may face while working in this area. Such a context places new requirements on management systems as they encompass the integration of service and network management. One key requirement is the need for modularity of management systems at an engineering level. This is seen as essential in order to obtain the level of implementation reuse and the resulting fast time to market that competitive pressures and changing customer requirements will demand in the open service

market. TMN recommendations offer the possibility for defining interfaces in a modular way, e.g. by using MFCs. However the mapping of modularity in interface specification to modularity in implementable components is outside the scope of TMN. This coupled with the slow emergence of an effective platform independent TMN API has meant that there is currently no commonly practised approach of modular TMN system design.

ODP, with its five view-points, supports implicitly the mapping modularity in the specification of functional interfaces to modularity in an engineering model. ODP has been taken by the TINA Consortium and applied to the telecommunications arena, though to date practical application of this architecture has been limited to a few prototypes. The work presented in this paper is an example of such a prototype, where some of the aspects from the TINA overall architecture have been applied to a home-grown DPE offering location transparency between engineering objects. By integrating this DPE with an existing TMN platform, the engineering aspects of applying ODP and TINA principles to TMN have been investigated to some extent. This is an important area, since even if TINA DPE or CORBA implementations are taken up increasingly for management applications there is a large investment in TMN-based management interface specification and an growing base of installed TMN agents and applications that will have to be accommodated by the new platforms. As outlined in this paper this involves both integration at a technological level, e.g. CORBA to TMN gateways such as that being developed for the ANDROMEDA TMN platform [Dittrich-95d], and at a design and engineering level. For the latter a clear mapping is required between TMN interfaces specified as MFCs and corresponding computational objects in specifying both interfaces to other systems (e.g. a TMN X interface) and interfaces between reusable components assembled in the same system.

The ACTS project, PROSPECT, will be building on the PREPARE and other related work in building further management networks for integrated multi-domain services in which an common engineering approach can be exercised. Use of CORBA-TMN gateway implementations is planned in investigating the needs of integrated network and service management in an open services market.

Acknowledgements

This work was conducted in close co-operation with all partners in the PREPARE project. In particular Sven Krause, Ingo Busse and others at GMD-Fokus are acknowledged for their part in applying the building block approach to their applications, for the design of some of the generic service components and for contributing to the concept of the DPE support for the building blocks. Lennart Bjerring of L.M. Ericsson and George Pavlou and David Griffin of UCL also deserve thanks for the important comments on this work. This work was conducted under the partial funding of the EC through the RACE II project PREPARE (contract R2004).

References

[Bjerring] Bjerring, L.H., Schneider, J.M., End-to-end Service Management with Multiple Providers, Proceedings of the 2nd International Conference on Intelligence in Broadband Services and Networks, Aachen, Germany, September 1994, pp 193-206, Springer-Verlag, 1994.

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

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

[M.3010] *Principles for a Telecommunications Management Network*, ITU-T Recommendation M,3010, 1992.

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

[NMF-025] *The "Ensemble" Concepts and Format, Forum 025*, Issue 1.0, Network Management Forum, Morristown, NJ, 1995.

[NMF-BPM] *A Service Management Business Process Model*, Network Management Forum, Morristown, NJ, 1995.

[NMF-SF] *The OMNIPoint Strategic Framework. A Service-Based Approach to the Management of Services and Systems*, Network Management Forum, Morristown, NJ, 1995.

[Q.1204] *Intelligent Network Distribution Functional Plane Architecture*, ITU-T Recommendation Q.1204, 1993.

[SPIRIT] *X/Open Consortium Specification. SPIRIT Platform Blueprint*, SPIRIT Issue 2.0, Network Management Forum, Morristown, NJ, 1994.

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

[TINA-018] Chapman, M. and S. Montesi, *Overall Concepts and Principles of TINA*, TINA Baseline document TB_MDC.018_1.0_94, February 1995.

[X901] *Information technology - Open Distributed Processing - Reference Model - Part 1: Overview*, ITU_T Draft Recommendation X.901/ISO/IEC Draft International Standard 10746-1, 1995.

[TCL-TK] John K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, 1994.

[IS&N95] D. Lewis, T. Tiropanis, L. H. Bjerring, J. Hall*, Methodologies for Multi-domain Management System Development*, IS&N '95 Conference in Crete, Greece, October 1995.

[TINA-012] H. Berndt, C. Kim, S. Kim, et al, *Service Architecture*, TINA Baseline document TB_MDC.012_2.0_94, March 1995.

[TINA-HV] N. Natarajan, F.Dupuy, N. Singer, H. Christensen, *Computational Modelling Concepts*, Baseline document TB_A2.HC.012_1.2_94, February 1995.

[OMT] J. Rumbaugh et al, *Object-Oriented Modelling and Design*, Prentice-Hall, 1991.

[Dittrich-95d] A. Dittrich, *The ANDROMEDA Platform: an object-oriented development and run-time environment for management services*, Proceedings of the 6th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM '95), 1995.

[CFS-H430] *The Inter-Domain Management Information Service*, RACE Common Functional Specification H430, RACE Industrial Consortium, Brussels, 1994.

[OSIMIS] G. Pavlou, G. Knight, K. McCarthy and S. Bhatti, *The OSIMIS Platform: Making OSI Management Simple*, in Integrated Network Management IV, ed. A.S. Sethi, Y. Raynaud and F. Faure-Vincent, pp. 480-493, Chapman & Hall, 1995.

[TCL-RMIB] T. Tin, G. Pavlou*, Tcl-MCMIS: Interpreted Management Access Facilities*, Proceedings of the Sixth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM '95), Ottawa, Canada, October 1995.

[Tcl-DP] B. C. Smith, L. A. Rowe, *Tcl Distributed Programming (Tcl-DP) (version 3.3beta1)*, ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/Tcl-DP, June 1995.

[OTcl] *Object Tcl (version b1.1)*, http://www.x.co.uk/devt/ObjectTcl/, IXI Limited, 1995.

Comment: This reference is taken from the PREPARE Book.