Applying UML to Design an Inter-Domain Service Management Application

Mohamed Mancona Kandé, GMD FOKUS (e-mail: Kande@fokus.gmd.de)
Shahrzade Mazaher, Norwegian Computing Center (e-mail: Shahrzade.Mazaher@nr.no)
Ognjen Prnjat, University College of London (e-mail: oprnjat@eleceng.ucl.ac.uk)
Lionel Sacks, University College of London (e-mail: lsacks@eleceng.ucl.ac.uk)
Marcus Wittig, GMD FOKUS (e-mail: Wittig@fokus.gmd.de)

Abstract

We present a component-oriented approach to demonstrate the use of the Unified Modeling Language (UML) and Open Distributed Processing (ODP) concepts to design service components of a telecommunications management system. This paper is based on the work done in the design phase of the ACTS project "TRUMPET" (Inter-Domain Management with Integrity). In our approach, we use the conceptual framework of ODP and discuss some methodological issues related to ODP-viewpoint modelling of distributed systems using UML notations.

Keywords: UML, ODP, ODP Viewpoints, object-oriented design, distributed systems, service provisioning, TMN, Java, CORBA.

1 Introduction

Designing and implementing complicated distributed control systems in large international groups or consortia is problematic. Techniques are required to ensure that all the participants understand where their work applies, the work of their collaborators, and the relationship between these. Further, when building systems which are expected to have some longevity it is important that people looking at the background documentation can understand what is involved. These considerations constitute important background to the use of architectural design methodologies and semi-formal methods in system modelling. A second important consideration is that the methodologies used facilitate close analysis of the design to ensure, at early stages, that the system is complete in meeting its requirements and consistent in its operation. This should yield robust, highly engineered products. Finally, when presenting the product to users, its functionality and roles within their business practices must be clearly documented and defined (following the motivations behind the NMForum OMINPoint Ensembles [NMF].

This paper illustrates how two popular methodologies can be combined to achieve these goals, and how this was done in the context of the TRUMPET telecommunications service architecture. The TRUMPET project undertook to produce a network service management architecture suitable for emerging liberalised telecommunications markets. The criteria where that the system should be highly distributed both technologically and administratively - *i.e.* across many kinds of organisations. Specifically, these organisations include: *Public Network Operators* (PNOs), *Customer Premises Network* centers (CPNs) and third party *Value Added Service Providers* (VASPs) or bandwidth retailers. Moreover, the TRUMPET consortium consisted of a number of engineers, from several companies across Europe who had to collaborate to design, build and run the system. Thus the

TRUMPET project presents a good model environment to develop not only the service architecture itself, but the methodologies for producing such designs.
The two methodologies used where the ITU-T Open Distributed Processing (ODP) architecture and Unified Modeling Language (UML). ODP is an architectural system which provides the designer of any distributed processing system a means of defining all the required elements. At the top level, ODP requires the designer to divide the design into a number of View Points'defined within the ODP Reference Model (RM-ODP) [ISO95]. In the design stage, the principal of these are the Enterprise, Information, Computational and the Engineering viewpoints. UML is a semi-formal software systems modelling language which provides a consistent diagrammatic semantics for systems design using mature object oriented philosophies. UML has derived from a number of other semi-formal modelling systems, most notably OMT, Booch, and OOSE. As with OMT, it has diagrammatic for defining objects, object relationships and object dynamics (internally with states & externally with collaborations). It augments OMT by providing very clear linkages between the occurrences of objects and methods in each of its diagrammatic structures. By using the appropriate UML diagrams for the appropriate viewpoints of ODP, it is possible to design a system with clear tractability between each of these models. UML & ODP, being well documented and highly accepted systems within industry, can be understood by engineers from many countries and companies. Thus using these methodologies together it is possible to present the designs in an - almost - universal languages between collaborators and to customers.

1.1 The TRUMPET Scenario

The ACTS project TRUMPET (Inter-Domain Management with Integrity) focuses on the secure operation of inter-domain management systems within the Open Network Provisioning (ONP) framework. The TRUMPET scenario in Fig. 1 involves the following players: two (or more) Public Network Operators (PNOs), a Value Added Service Provider (VASP), and a number of customers at various sites - Customer Premises Networks (CPNs) [D6] [TSMA].
The customers see an end-to-end connection and are not necessarily aware of which PNOs are contributing to establish the connection. The VASP sees the connection as a set of *segments*, each supported by a different PNO, but does not know how each segment has been set up within the corresponding PNO (*i.e.*, what ATM switches are used).
The management systems of the players mentioned above form a service provisioning system for management and provision of broadband (ATM) network connections between two customers/end users. CPN is a dedicated service in the customer organisation, which already has a contract with the VASP. The VASP management system provides network connectivity to customers by utilising the resources of one or more PNOs. VASP allows customers to create, modify and delete connections, thus effectively providing the Virtual Private Network (VPN) service to the customers. PNOs provide the physical infrastructure, i.e. the network, and the adequate management interface to interact with it.

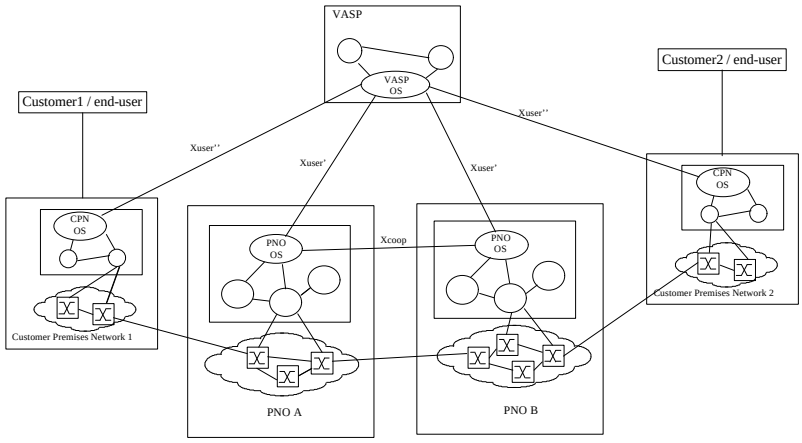

Fig. 1: TRUMPET Scenario

The TRUMPET management system thus represents a typical component-based application for distributed service management. Entering the design phase, a need for suitable methodology and supporting notation for modelling such a distributed system emerged. Having critically reflected on existing methodologies and notation schemes, the consortium decided to adapt the UML concepts to the ODP framework and design the system in such a fashion.

1.2 UML and ODP

The approach to mapping the different ODP-viewpoints to UML diagram types as depicted in Fig. 2 [KTW97] has been adopted in the TRUMPET project. The major benefit of such a mapping consists in supporting both an object-oriented modelling and design process, and the design of reusable components and distributed services [JAC97] in the sense of distributed systems as described in the *Reference Model for Open Distributed Processing* (RM-ODP) [ISO95]. ODP defines clearly the concept of viewpoints, and it provides several different viewpoint languages, which may be unified using UML notations and concepts for specifying all these viewpoints. Thus, an integration of ODP and UML allows for the efficient modelling of the ODP viewpoints, supporting the provision of a new framework for the design of distributed management systems and the specification of the internal and external behaviour of components. Such integration also helps to map and to implement some ODP functions in different technologies like CORBA and TINA-DPE [TINA].

Fig. 2 depicts the links between ODP viewpoints [BB96], and shows the relationships between these viewpoints and the UML diagrams, as well as between UML diagrams themselves (which are mapped to the same viewpoint specifying different aspects of same objects). The design of a distributed management application may start with capturing requirements of the system in terms of *Use Case Diagrams*. The results obtained from a use case model may be used to present high level *Static Structure Diagrams* as indicated in the *Enterprise Viewpoint*, and these diagrams can be specified in more detail in the *Information Viewpoint*. The step from the *Static Structure Diagram* to the *Statechart Diagram* allows to specify the dynamic behaviour of significant information objects.

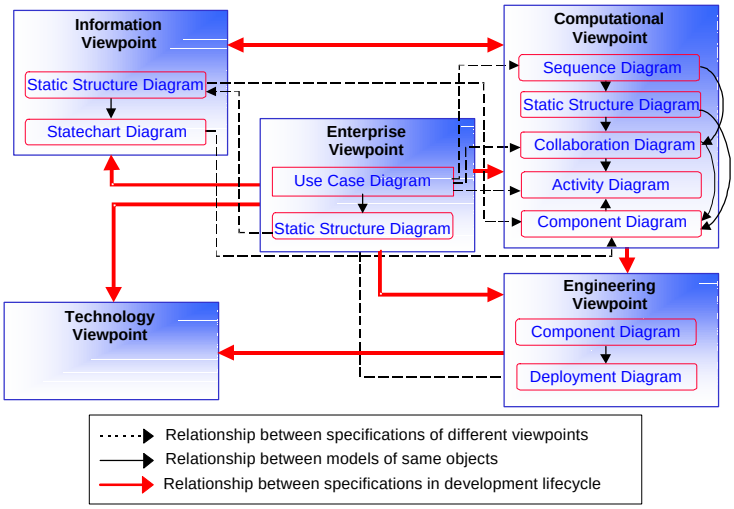

Fig. 2: Relationships between UML Diagrams and ODP Viewpoints

The *Computational Viewpoint* maps to the *UML Collaboration*, *Sequence*, *Activity* and *Component Diagram* types. While the UML *Collaboration Diagram* shows the interactions among instances and their links to each other, the *Sequence Diagram* describes object interactions arranged in a time sequence. The *Activity Diagram* allows to specify in which order activities (such as operations provided by a computational object interface) have to be executed. The *Component Diagram* shows the organisations and dependencies among components.

In addition to the *Component Diagram*, the *Deployment Diagram* is also mapped to the *Engineering Viewpoint*. This diagram type shows how components and objects are routed and moved around the distributed system [FOW97]. There is no mapping between UML and the *Technology Viewpoint* offered by this approach.

2 Case Study

This section describes how, within the ODP framework, different UML concepts and notations were used to design the TRUMPET management system. A separate section is dedicated to each of the five ODP viewpoints by projecting the VPN service in the corresponding viewpoint using the appropriate UML notation schemes.

Each viewpoint first gives an overall view of the TRUMPET management system, and then focuses on a more detailed description of the VPN service within the VASP domain.

2.1 Enterprise Viewpoint

The ODP Enterprise Viewpoint represents an overview of the systemś aims, constraints and functionality as seen by the enterprise. This Viewpoint models the basic system decomposition into components, identifies actors, policies and domains, and describes the general scenarios of the systemś use.

The TRUMPET system incorporates three domains (or enterprise objects in ODP terminology) namely the CPN, the VASP and the PNO. The PNO domain is further sub-divided into PNO Service Layer (SL) and PNO Network Layer (NL). These four

entities were modelled as UML Packages with interdependencies, using UML Static Structure Diagram notation as shown on Fig. 3.

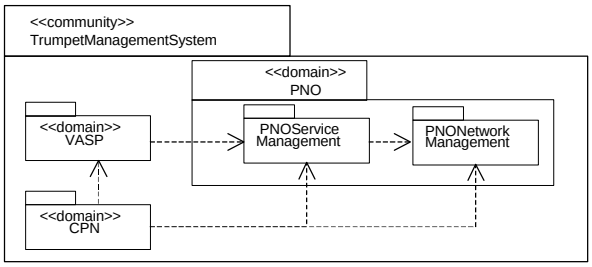

Fig. 3: The TRUMPET Static Structure Enterprise Packages

Next, the desired functionality of the system were described. In the ODP context, this is done by specifying the scenarios, or Use-Cases, which describe how actors/entities interact in the context of using the system.

The TRUMPET scenarios were specified using the UML Use-Case diagram, depicting the actors (human users), sets of Use Cases (ellipses) within a system, and associations between actors and Use Cases - illustrated in Fig. 4. The solid lines between the User and the Use Cases represent the „communicates“ relationship, denoting the participation of the user in the Use Case. The lines with arrows between Use Cases represent the „extends“relationship - indicating that the terminating Use Case may include the behaviour specified by the originating Use Case.

Note that the UML stereotype *<<community>>* has been used to classify the high-level enterprise object *„TrumpetManagementSystem“* as community in the sense of ODP.

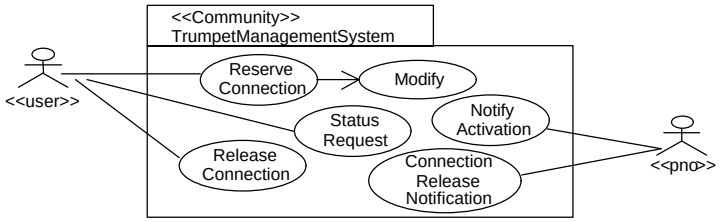

Fig. 4: The Use-Case Diagram

Fig. 4 depicts the functionality (or Management Functions) identified in the VASP management service as Use-Cases, and the interaction of the different users with the Use-Case package. As shown, there are six Use Cases. Customers/end users are capable of reserving end-to-end connections (of a given duration and desired Quality of Service (QoS)), modifying them (changing duration, QoS, or both), and releasing, i.e. deleting these connections. PNOs are capable of notifying the users via the VASP of connection activation, or notifying the users of the connection release due to a segment/link failure.

2.2 Information Viewpoint

In the ODP Information Viewpoint, the information objects of the system are identified and their structures and relationships described. UML *Static Structure Diagrams* were used to describe the static structure of the information objects.

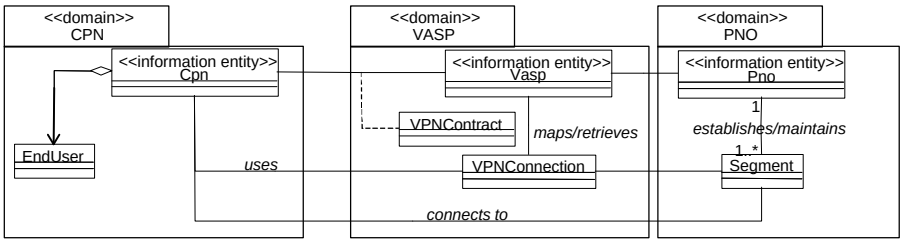

Fig. 5: The Static Structure of the VPN Service

Fig. 5 illustrates, by means of a UML Static Structure Diagram, the overall structure of the VPN service. That is, what entities are involved, what their relationships are, and how they interact.

In the following, the emphasis is put on the VASP entity of the above diagram, and a more detailed description of this entity is given. In the static structure diagram of Fig. 6, the VASP is decomposed into its three main components. The *CustomerServer* handles the communication with the customer domain (CPN). The *ControlServer* which, after negotiations with the involved PNOs, establishes, modifies, or releases the VPN connections; and the *MIB* (Management Information Base) which contains all the Managed Objects (MO). These objects contain information about the different entities that the VASP either interacts with or manages. For example, objects containing information about the VASP customers or objects representing the connections that the VASP manages, are all contained in the MIB. Both the *CustomerServer* and the *ControlServer* have the access to the MIB either by retrieving information from it or updating it.

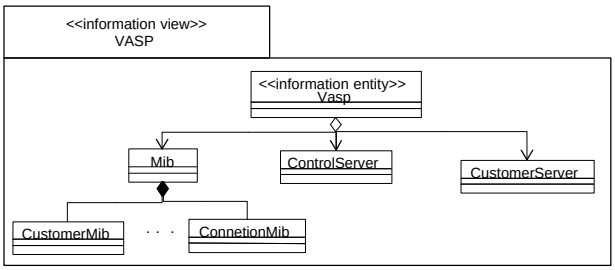

Fig. 6: The Overall Static Structure of the VASP

Among the different components of the VASP the MIB represents the pure informational objects. The two other entities are rather information processing units (manipulating the MIB) although each of them contains information about the current state of the VASP. In the rest of this section, UML static structure diagrams are used to give a more detailed description of the MIB.

The customer MIB contains information pertaining to the VASP customers, their respective service profiles, and the terms of their subscriptions. A corresponding

MIB exists for the PNOs the VASP is dealing with. These MIBs are rather static, in the sense that the information they contain is seldom updated. The Connection MIB contains information about all the connections that the VASP is currently supporting. Furthermore, its structure reflects the view that the VASP has of a connection, i.e. a connection consisting of segments individually supported by a PNO. This MIB is being constantly updated (therefore dynamic) as requests for new VPNs and change/release of the existing ones are received from the customers.

2.3 Computational Viewpoint

The Computational Viewpoint describes how the management functions, identified by means of enterprise Use Cases, are performed by the management system. Each management function is described in terms of computational objects (COs) and computational activities, the latter representing sequences of operations invoked on COs, e.g. components.

As a starting point for the computational design, the components identified in the Enterprise Viewpoint can be mapped to COs which provide an abstract, course grain computational view of the management system. Each component can then be broken further down into a set of COs which represent the detailed computational object model. At this level, the UML static structure diagrams have been used to describe the structure of COs, their multiple interfaces, and the relationships between them. At the abstract level, UML component diagrams have been used to describe the system components and their external interfaces.

Fig. 7 depicts the design of the *VaspVpnManager* component (referred to in the Information Viewpoint as the *Control Server).* We distinguish the *VaspVpnManagerFacade* package containing the external structure (client view of the component) from the *VaspVpnManagerIml* package which contains the implementation details of the internal class structure.

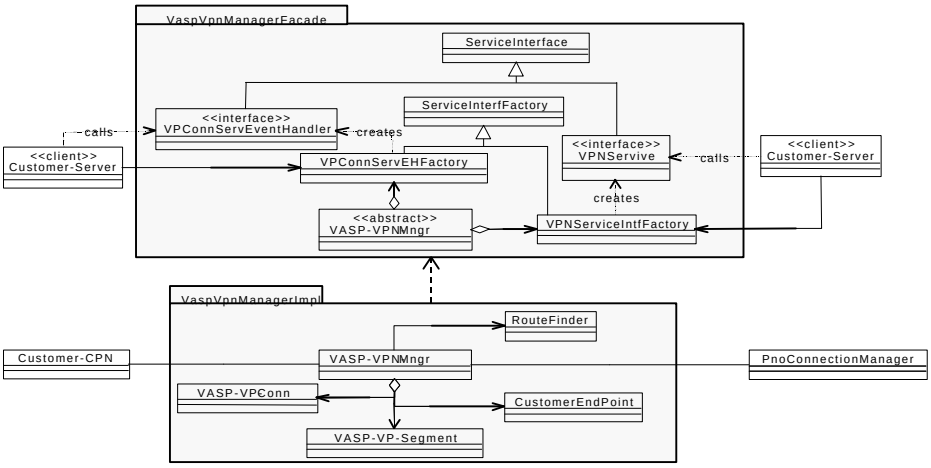

Fig. 7: Internal Structure of the *VASP-VPN-Manager* Computational Object Type

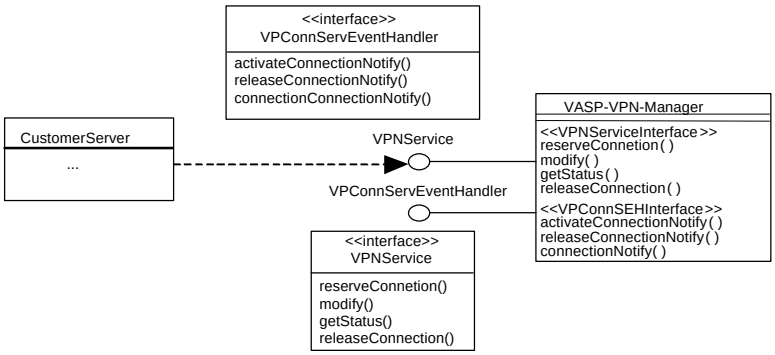

Fig. 8: The *VASP-VPN-Manager* Computational Object Type Diagram with Interfaces

The *VaspVpnManagerIml* package realises the interfaces contained in the „façade" package as well as other internal objects helping to realise the management functions supported by the computational component interfaces. The structure of classes as shown in the *VaspVpnManagerFacade* package depicted above may be used to provide the information necessary to define an OMG-IDL file. Thus, such a file may be more easily written using computational object and interface type diagrams as a basis (Fig. 8). In IDL these interfaces might be defined in a file *„VpnManager.idl"* which has the following form:

```
#include "XuserTypes.idl"
module VpnManager {
    interface VPConnectionServiceEventHandler {
        oneway void activateConnectionNotify(
            in XuserTypes::NameType pnoId,
            in XuserTypes::NameType vpConnectionId,
            in XuserTypes::ActivationNotifInfoType status);
        oneway void releaseConnectionNotify(
            in XuserTypes::NameType pnoId,
            in XuserTypes::NameType vpConnectionId,
            in XuserTypes::ReleaseReasonType reason);
        oneway void connectionNotify(in XuserTypes::NameType pnoId,
            in XuserTypes::ReasonType reason,
            in ASN1_PrintableString eventInformation);
    };
    interface VPConnectionService {
        exception ConnectionRequestFailure {
            XuserTypes::ReasonType reason;
        };
        XuserTypes::ReserveConnectionResultType reserveConnection(
            in XuserTypes::ReserveConnectionInfoType connectionInformation)
            raises(ConnectionRequestFailure);
        ...
    };
}; // End of Module
```

In the above case, the file *"XuserTypes.idl"* contains all common definitions used for data types.

Next, the computational activities are described. Computational activities depict the interactions between the COs in order to perform the management functions defined through Use-Cases in the Enterprise Viewpoint. Interaction between COs is described in terms of an operation invocation initiated by a client object requesting an operation to be performed by a server object. Precedence rules are used to define the sequence of operations performed when an interaction takes place. To describe the COs' internal and external interactions (between different COs), UML collaboration diagrams and sequence diagrams have been used.

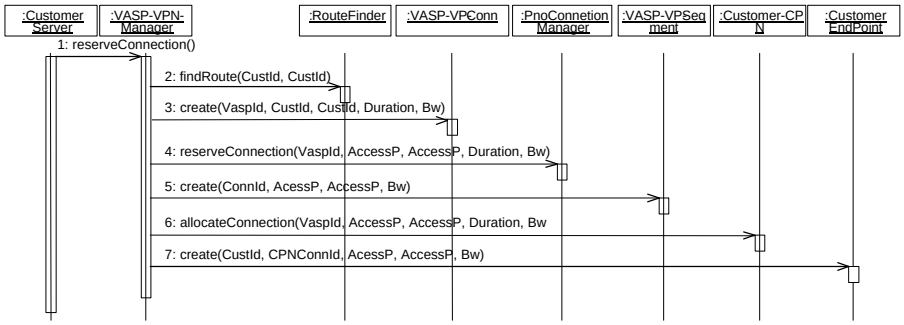

Fig. 9: The *Reserve Connection* Sequence Diagram

The UML Sequence and Collaboration Diagrams as depicted in Fig. 9 and Fig. 10 describe the *Reserve Connection* Use Case defined in the Enterprise Viewpoint. These diagrams depict the interactions, sequences of messages, and relationships among computational components (such as PnoConnectionManager, VASP-VPN-Manager) as well as programming level objects (e.g. instances of UML object within the *VaspVpnMamager* package) [KSW97].

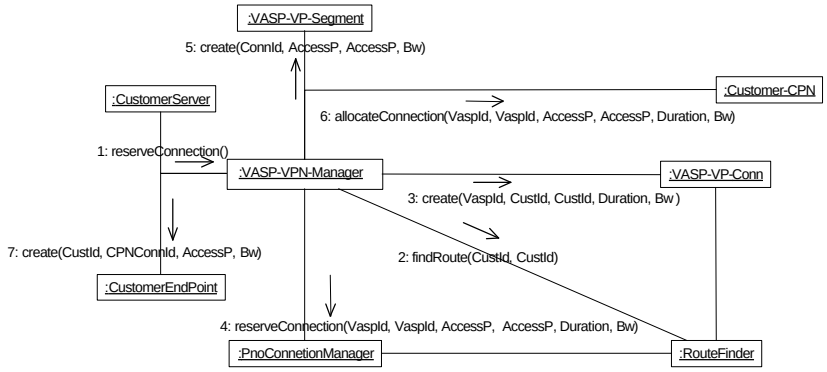

Fig. 10: The *Reserve Connection* Collaboration Diagram

2.4 Engineering Viewpoint

This viewpoint focuses on the actual realisation of interactions between distributed objects and on the resources needed to accomplish this interaction. It comprises of concepts, rules and structures for the specification of the system viewed from the engineering perspective.

Within this viewpoint, the engineering specification uses the *UML Deployment and Component Diagrams* to define mechanisms and functions required to support distributed interactions between objects in the service management system.

As depicted in Fig. 11, the concept of *UML-nodes* was applied to design *PNO-Host*, *VASP-Host*, and *CPN-Host*. Each of the UML-components contained in the nodes (such as VpnManagerServer and PNOConnMngrServer) represent a configuration of engineering objects forming a single unit for the purpose of encapsulation of processing and storage. Such a configuration is called a *Capsule*. The Capsule (Customer Server) contains a composite object mentioned by the stereotype

<<*cluster*>> that may migrate from the VASP-Host to the CPN-Host as indicated below by <<*becomes*>> stereotype.

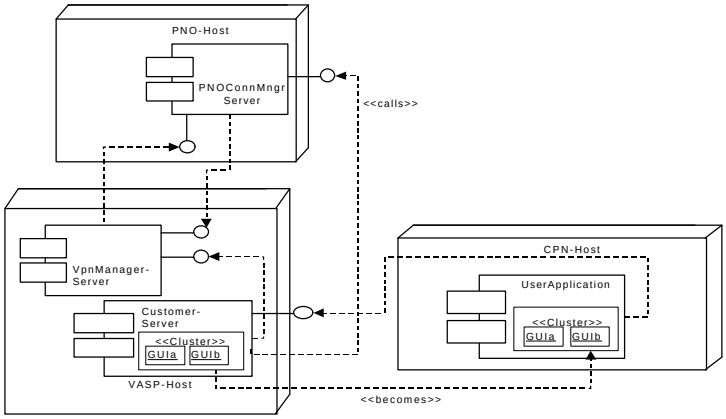

Fig. 11: Deployment Diagram

2.5 Technology Viewpoint

The ODP Technology Viewpoint describes the choice of implementation technologies used to bring the design accomplished through the four previous viewpoints to life. This viewpoint depicts the configuration of the hardware and software on which the distributed system relies. Although there are no dedicated UML diagrams to describe this viewpoint, the Object Oriented concepts of encapsulation and abstraction provided by UML in previous design steps allow the system to be implemented on a mixture of computer architectures, programming languages and operating systems. This is one of the big advantages of the OO approach adopted by UML. Thus, the Technology Viewpoint was described using plain English.

The CPN is realised as a group of Java objects providing an interface to the VASP VPN functionality. A HTML based user interface is also provided as the end-user interface to CPN Java objects. The CPN-VASP communications are implemented in Voyager, the Java-based communications mechanism. VASP is fully implemented in Java, apart from the MIB, which is based on the Lightweight Directory Access Protocol (LDAP) which effectively supports functionality required by the TMN-like MIB. PNO management system is a TMN-OSI platform, the HP-OV, which communicates via CMIP protocol. Thus, VASP implementation requires a JAVA-to-CMIP gateway, which is realised as a platform-independent CORBA gateway. These technologies were chosen so as to fulfil the trial requirements and project aims.

3 Conclusion

The RM-ODP defines, as mentioned before, five different viewpoints and the core concepts of each one, used for the description of distributed systems. However, it leaves the issue of how to express each viewpoint, i.e. the notation to be used, open. In this paper we discussed the use of UML notations in the context of the ODP framework for the design of telecommunications management services.

Combining UML and ODP has several advantages. First, both UML and ODP are based on the object paradigm. There is therefore no conflict of basic concepts between the two.

The various diagram types of the UML notation, such as sequence diagram, collaboration diagram, etc., make it possible to describe the same system from different perspectives. This matches rather well with the RM-ODP different viewpoints as shown by the example used throughout the paper - appropriate types of UML diagrams are mapped to the different ODP viewpoints. Using the same notation for all the viewpoints has also the advantage of making both the system description more coherent and the process of shifting from one viewpoint to another more natural and smooth - one only shifts context. Conversely, RM-ODP framework proved to be an efficient way of structuring different UML notations and thus managing the potential high complexity introduced by various UML diagram types.

Although the UML notation is an attractive choice for use in the design of a distributed system, it also has some drawbacks. Some of the ODP concepts are not directly supported by UML. For such reasons, UML introduces the concept of stereotypes to provide for extensibility. In the example of the previous sections, stereotypes have been extensively used to map RM-ODP concepts that did not have a direct counterpart in the UML notations, such as enterprise objects, communities, clusters, etc. Moreover, although the concept of an interface is part of UML, its description uses the same notation as for a class. Again a stereotype, <<*interface*>>, has been used to differentiate between the class and the interface descriptions. The use of the same notation to express quite different concepts becomes at best confusing. In a pictorial notation the core entities of a model (e.g. RM-ODP) should have individual representations as to be readily distinguished from one another.

Also, UML did not prove to have enough power to fully describe the ODP concept of the Computational Object. During the design, only the external interfaces provided by a component were specified, and concepts like binding rules and lifetime aspects were not included.

In conclusion, the use of the ODP-UML methodology in TRUMPET proved efficient in collaborate work, resulting in coherent and detailed design of the distributed, component based inter-domain service management system.

ACKNOWLEDGEMENTS

This paper is based on original work developed by the ACTS project TRUMPET. The authors wish to thank all the partners of the TRUMPET consortium who contributed to this work. More information on the TRUMPET project can be found at http://ascom.eurecom.fr/ASRL/TRUMPET/Trumpet_public/.

4 References

[KSW97] M. M. Kandé, S. Tai, M. Wittig, *On the Use of UML for ODP-Viewpoint modeling.* In OOPSLA 97 Workshop on Object Oriented Technology for Service, System and Network Management, Atlanta, Geogia, U.S.A., October 1997.

[FOW 97] Martin Fowler with Kendall Scott, *UML Distilled: Applying the Standard Object Modeling Language*. Addison-Wesley, 1995.

[JAC97] Ivar Jacobson, Martin Griss, and Patrik Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success*. Addison-Wesley,1997.

[OMG 97] Rational Software, Microsoft, Hewlett-Packard, Oracle, Texas Instruments, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp. *The Unified Modeling Language*, Joint Submission, OMG TC doc ad/97-01-01 - ad/97-01-14 .

[ISO95] ISO/IEC 10746-1/2/3. *Reference Model for Open Distributed Processing -Part1:Overview/Part2: Foundations/Part3: Archictecture*, ISO/IEC, 1995.

[BB96] Kim Berquist, Andrew Berquist (eds.). Managing Information Highways. *The PRISM book: Principles, Methods and Case Studies for Designing Telecommunications Management Systems*, Springer LNCS 1164, 1996.

[HAL 96] J. Hall [Ed.] , *Management of Telecommunication Systems and Services: Modeling and Implementing TMN-based Multi-domain Management*. Berlin; New York; Tokyo: Springer Verlag, 1996.

[TINA] Graubmann, P. and N. Mercouroff, *Engineering Modelling Concepts* (*DPE Architecture*), TINA Baseline document TB_NS0005_2.0_0.94, December 1994.

[TM] T. Mowbray and R. Malveau, *CORBA Design Patterns*, John Wiley and Sons Inc, New York, USA, 1997.

[D8] ACTS Project AC112 TRUMPET Deliverable 8, *Detailed Component and Scenario Designs*, June 1997.

[D6] ACTS Project AC112 TRUMPET Deliverable 6, *NIL-Security Prototype Report*, February 1997.

[NMF] *The OMNIPoint Strategic Framework. A Service-Based Approach to the Management of Network and systems*, Network Management Forum, NJ, 1993.

[TSMA] L. Sacks et. al., *TRUMPET Service Management Architecture*, submitted for IS&N, 1998.