Modeling the Architecture of Distributed Real-Time Systems

Dominikus Herzberg*
Ericsson Eurolab Deutschland GmbH
Ericsson Allee 1
52134 Herzogenrath, Germany
Dominikus.Herzberg@eed.ericsson.se

André Marburger
Aachen University of Technology
Department of Computer Science III
52074 Aachen, Germany
marand@i3.informatik.rwth-aachen.de

Abstract

ROOM (Real-Time Object-Oriented Modeling) and its UML derivate, UML-RT (Unified Modeling Language for Real-Time), are common and relatively popular languages for modeling distributed real-time systems. However, both languages provide only limited support for higher level structural organization of models. In data and telecommunications, systems are typically structured into layers, planes, and distributed peer-to-peer relations. Corresponding structural concepts are almost missing in ROOM and UML-RT. This report presents in an overview style, how a generalized port concept provides the means to introduce so-called Service Access Points *and* Connection Endpoints *as modeling concepts to ROOM/UML-RT. It allows a modeler to precisely describe any distributed communication architecture that is structured in layers and planes. In addition to this, the notation presented puts us into a position to define and identify architectural patterns.*

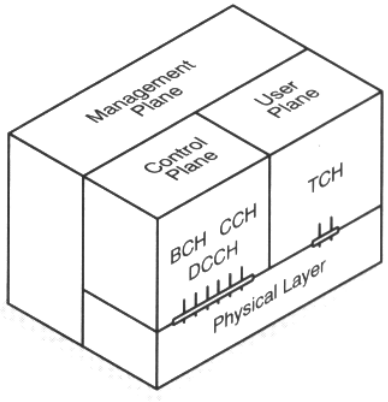

Figure 1. Informal model of the ISDN/GSM system architecture; taken from [6, p.117]

1 Introduction

Systems designers of the data and telecommunication domain are still reluctant to use modeling languages like the UML (Unified Modeling Language) [16] for describing their system architectures; the UML only slowly infiltrates the systems engineering domain [7]. One reason is that today's modeling languages simply do not support the kind of high-level organizational constructs systems designers are used to. In practice, architectural descriptions are often informal documents [8]. The lack of a proper notation and formalism very often leads to diagrams similar to figure 1. Figure 1 shows a simplified version of the GSM (Global System for Mobile communication) plane architecture, which is almost identical to the ISDN (Integrated Services Digital Network) architecture. Such diagrams provide at best a basic view on the architectural conception rather than a precise architectural model.

In the datacom and telecommunication domain, some few concepts are vital for architectural design. This report shows that the study of such domain specific concepts can be fruitful to the design of real-time systems [12]. In particularly, we investigate the telecommunication domain, which is by nature distributed and real-time, and focus on the concepts of *layers* and *planes*, and on the *relation of layers* of distributed entities. Our main reference of authority for the telecommunication concepts is the Open Systems Interconnection Reference Model (OSI RM) [15], which has laid a solid foundation for understanding distributed open system intercommunication [2]. Apart from its didactic value, OSI RM manifests many concepts and principles, which are still in use and have influenced the design of today's communication systems. Further information and commentary about OSI RM can be found e.g. in [17, 21]

*This work is being funded by Ericsson and run in cooperation with the Department of Computer Science III, Aachen University of Technology, Germany.

Two prominent modeling languages enabling the design of distributed real-time systems are ROOM [18] and UML-RT [19, 20]. UML-RT is, so to speak, an adaptation of ROOM's notation to UML and defines a profile for UML. For this study, we have chosen ROOM/UML-RT as a candidate for modeling communication systems [10]. However, neither ROOM nor UML-RT provides sufficient architectural support for structuring these systems. Therefore, a semantical and notational extension to ROOM/UML-RT will be proposed in order to capture the concepts identified by studying OSI and ISDN. It can be shown that specializations of a generalized port concept would suffice as a language extension.

In section 2, the concept of *Service Access Points* is introduced, which can be used to structure a system into layers and planes. The concept of *Connection Endpoints* is the subject of section 3; it is required to indicate communication relations of distributed peer entities. Finally, section 4 closes with some conclusions. Note that this paper intends to give only an overview of our approach to architectural modeling rather than a detailed discussion. More information can be found in [11].

2 Modeling Layers and Planes

2.1 The Concept of Service Access Points

The concept of *layers* is a key characteristic of all communication systems. It is a means of stepwisely increasing the degree of abstraction and of separating levels of abstraction by precisely defined interfaces, and is reflected by the use of protocol stacks. The OSI RM is characterized by a layered architecture, which is not repeated here. Almost any textbook on computer networks and/or data communications gives an introduction into OSI RM, for example [9, 21]. For the purpose of this discussion, we have limited ourselves to the part of OSI RM that concerns the separation of one layer from another. This selection restricts the considerations on the cooperation of two arbitrarily selected but adjacent layers.

Layering is a form of information hiding. A lower layer presents only a service interface to an upper layer, hiding the details of how it provides the service. Layers present different levels of abstraction on system functionality in a stepwise manner. Closer investigation into the principle of layering highlights the *Service Access Point* (SAP) as a key concept, which has been introduced by OSI RM. The SAP is owned by the *service provider* (the "lower" layer) and provides services by means of service primitives to a *service user* (the "upper" layer). The SAP not only specifies an interface by a dedicated set of so-called service primitives and rules; it is also associated with further attributes relevant to communication systems, such as an identifier for addressing, SAPI (SAP Identifier), and Quality of Service (QoS) properties. In principle, SAPs are also less restrictive than layering is, they allow the designer to violate stacking of layers; this is an option and reflects the needs of design in practice.

2.2 An Extension to SAPs

A concept that OSI RM lacks (which is also one of its major deficiencies) is the concept of a *plane*. The concept was introduced in ISDN (Integrated Services Digital Network) [13], taken over in GSM (Global System for Mobile communication) [6], and currently shapes the network architecture of UMTS (Universal Mobile Telecommunications System). The distinction is usually in three planes, namely the *control plane*, the *user plane*, and the *management plane*, see figure 1.

A plane encapsulates service functionality and may have internally a layered (protocol) structure. Planes are an organizational means on top of layering. In telecommunications, the *user plane* with its layered structure provides for user information flow transfer, along with associated controls (e.g. flow control, recovery from errors); the *control plane* with its layered structure performs call and connection control functions, dealing with the necessary signalling to set up, supervise, and release calls and connections; the *management plane* takes care of (a) plane management functions related to the system as a whole including plane coordination and (b) functions related to resources and parameters residing in the layers of the control and/or user plane [14]. Figure 1 displays the relation of the planes in a three-dimensional arrangement; the *physical layer* is shared by the control and user plane for transmission purposes. TCH, BCH, CCH, and DCCH refer to channels, which are out of the scope of this discussion.

Even though the designers of ISDN might never have thought about it, a simple extension of the SAP concept makes it applicable for capturing planes as well. If we add means to distinguish SAPs from each other, we are also capable of distinguishing planes. A classification of SAPs e.g. by attributing SAPs allows us to clearly identify an SAP as a member of a set of SAPs, which is just another paraphrase for planes. The idea is easy to understand if visualized, see figure 2. To fully comprehend the figure, the following conventions for an SAP notation are introduced; this is also our notational proposal for ROOM/UML-RT.

2.3 A Notation for the Extended SAP Concept

The preferred notation for an SAP is the port symbol rotated by 45 degrees, which results in a diamond symbol. The counterpart of the SAP, the SAP^{-1}, is visualized by an "empty" diamond. Note that the SAP^{-1} is different from

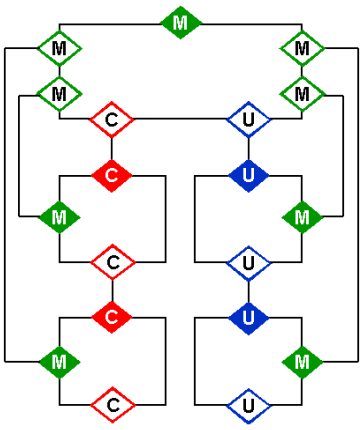

Figure 2. Precise architectural model based on ISDN, fulfilling figure 1

the SAP: it contains neither address nor QoS attributes. The SAP^{-1} just indicates that the capsule is prepared to connect to a service provider. The SAP and its complementing SAP^{-1} are identified by their interface protocol descriptions P and P^*. (The conjugated protocol P^* has the same definition as P except that the incoming and outgoing message sets are interchanged [18, p.154].) One can only interconnect SAPs and $SAPs^{-1}$ that have the same expectations on their interfaces. That means that numbering layers is now redundant, though it may be convenient to do so. It is, of course, forbidden to connect SAPs with SAPs and $SAPs^{-1}$ with $SAPs^{-1}$.

SAPs are a port-like concept, but they are semantically different. The similarity is that SAPs and ports can (but do not need to) be based on an homogeneous communication model of interaction (e.g. message based protocols) [18, p.200]. The difference is that SAPs are a structuring measure, they organize collectives of capsules (or actors) communicating via ports into layers. That means that we have to introduce a semantic rule saying that if two peer-to-peer networks are separated by an SAP/SAP^{-1} pair, none of the capsules is allowed to be connected to another capsule of the other network via ports. In other words, peer communication should not bridge layers. The SAP/SAP^{-1} concept constraints how capsules may communicate to each other via ports. It is this constraint that semantically introduces peer-to-peer and interlayer communication. In ROOM, it is possible to violate that semantic rule [18, p.208].

Planes extend the SAP concept and its notation. Planes are distinguished by a name and notated by the first letter (in capital) specified by the plane name inside the diamond symbol of the SAP and the SAP^{-1}, respectively. This is the default notation if no other character or string is chosen, and if no conflicts appear due to identical first characters of the plane name. Otherwise two or three letter codes may be used. If there is only one plane in the model, no character is required at all. Optionally, a color code may be associated with the plane name coloring the diamond symbol. Black is the default color. It is recommended to use both the color code and the centered capital inside. This makes models easier to read on e.g. a color monitor, but preserves the information if printed out in black and white. The following plane names have a predefined associated notational character and color code: `U` and color "blue" identify the *user plane*; `C` and "red" the *control plane*; `M` and "green" the *management plane*. If the colors available are used up, "black" becomes the default color.

2.4 Examples of Architectural Models

Taking a closer look now at figure 2 reveals that it is a feasible but concrete and precise architectural realization of the informal diagram in figure 1. We can identify two planes, a user and a control plane, each with two independent layers. For the sake of brevity, the access to the physical layer is not shown. The capsule at the very top provides a management SAP to a service user, and accesses the user and the control plane; the capsule fulfills the management function of plane coordination like the *Synchronization and Coordination Function* does in ISDN. In addition to that, each capsule of the user and the control plane provides a management SAP, which allows to access plane/layer specific resource and parameters functions (the other aspect of the management plane). With the upmost capsule as a shared layer resource included, we can count three layers in the user plane, three layers in the control plane, and two layers in the management plane.

Another possible architectural solution is shown in figure 3. Here, there is still a coordinating capsule at the very top, but there are three interconnected planes: Capsules of the management plane have a peer-to-peer server/client relationship to capsules of the control plane (remember, "white" ports mark servers), and capsules of the control plane have an analogous relationship to user plane capsules. This solution requires that all planes have the same number of layers. Given that the server capsules have the capability to incarnate and destroy their clients (ROOM provides a notation for so-called *optional* actors), we end up with a highly dynamical architecture in which the management plane composes the control plane, which in turn composes the user plane. This architecture in fact is a lightweight realization of the *Modular Communication Systems* (MCS) reference framework as described in [3].

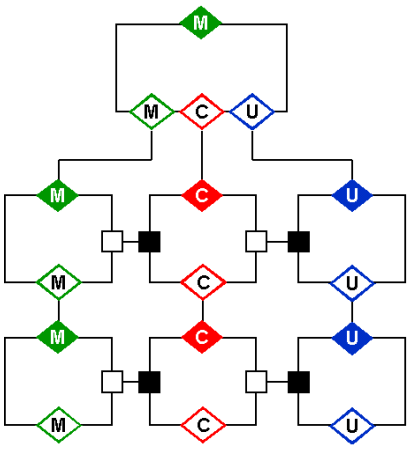

Figure 3. Precise architectural model based on the MCS framework, fulfilling figure 1

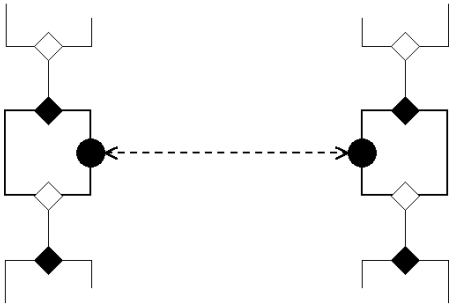

Figure 4. A simple inter-layer communication model

3 Modeling Protocol Relations of Distributed Peers

3.1 The Concept of Connection Endpoints

SAPs "vertically" structure a communication systems; they separate layers and slice them into planes. In the "horizontal" dimension there is the exchange of information between remote peers. Remote peers are physically distributed, they reside in different nodes, and communicate with each other according to a protocol. The "point" describing the protocol interface is called *Connection Endpoint* (CEP), which can be uniquely identified by a *Connection Endpoint Identifier* (CEPI) [15].

Note that the term "protocol" has a refined meaning in data and telecommunications. Even though the concept of a protocol is generally defined as a set of messages and rules,[1] software engineers assume a reliable, indestructible communication relation in their software systems, whereas data/telecommunication engineers have to face the "real" world: they have to add error correction, connection control, flow control and so on as an integral part to the protocol. A communication relation between remote peers can always break, be subject to noise, congestion etc.This is the reason why communication engineers introduced protocol stacks, with each protocol level comprising a dedicated set of functionality, thereby "stackwisely" abstracting the communication service. These stacks naturally give means to "vertically" dividing a node into layers.

[1]Take e.g. the definitions given by [21, p.27] as a representative of the data communication camp and [1, p.191] as a representative of the software engineering camp.

3.2 A Notation for the CEP Concept

The CEP is symbolized by a filled circle and denotes a potential interaction/connection point to a remote peer. The actual relation between CEPs is shown by a connecting line with arrow heads indicating the direction of communication, see figure 4. If the connection relation is *logical*, the line is dashed; if the connection relation is for *"real"* (concrete) the line is solid.

The distinction of the communication relation into *logical* and *concrete* points out the somewhat hybrid nature of CEPs. In fact, the CEP has two functional purposes: (1) Without accessing any provisioning services of a lower layer, the CEP is needed to describe and possibly simulate node interaction; this is a purely *logical* view of protocol layer interconnection. In this view, the CEP is connected to a logical entity modeling the channel connection including transmission characteristics. (2) On the other hand, if the provisioning layer is attached, the SAP^{-1} substantiates or implements the CEP. The CEP becomes "inactive" and its function is taken over by the SAP^{-1}. The CEP may just indicate, which protocol messages are virtually being sent out, it no longer plays an active role.

This dichotomy is also known as *refinement* [4, 5] and it is not a trivial exercise to properly reconcile both aspects in a single model.

3.3 An Architectural Pattern

A possible solution to the hybrid CEP problem is presented in figure 5. If the capsule holding both the SAP and the SAP^{-1} is divided into two parts, we can put a CEP pair in between that is non-virtually connected. This solution is compatible with OSI RM. According to OSI, the SAP "owns" the CEP; here, it is the "upper" capsule owning both the SAP and the CEP. Note that the "empty" CEP fulfills the same function as does SAP^{-1}; protocols can also be defined

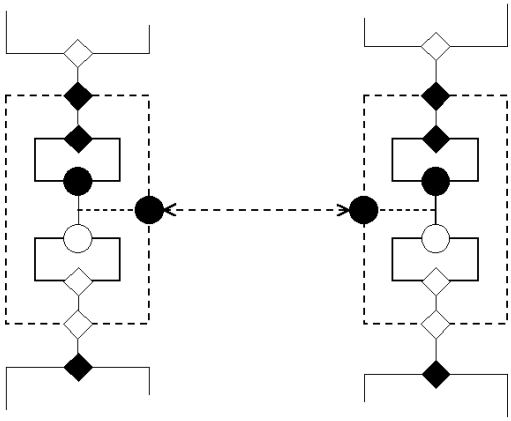

Figure 5. An architectural pattern: Integrating CEPs in architectural models

correspondingly. If required, the pattern can be embedded in a "higher-level" capsule exporting the inner interfaces.

The pattern shows that the layer in the communication model specifies a remote peer protocol and that it has an addressable communication point for that; it makes the protocol specification explicit instead of hiding it inside a capsule. However, how the protocol is finally implemented is a different story. For a modeler of distributed real-time systems it is an important aspect to show protocols and their reference points on an architectural level; it is relevant information when it comes to modeling communication networks of distributed entities.

4 Conclusions

This paper gave only an overview of our approach to architectural modeling of distributed real-time systems. Many details were neglected since we did not intend to provide an in depth study, but rather, demonstrate that a variety of architectural solutions can be described with the CEP concept and the extended SAP concept. A thorough discussion about layers and planes is given in [11]. The preciseness achieved is a significant improvement over informal descriptions, and enables system designers or architects to uniquely specify and describe the organizational structure of a distributed real-time system into layers, planes, and distributed peer communication. The extended SAP concept can be used to describe the architectural design of "traditional" solutions, like ISDN, as well as quite modern approaches such as the MCS framework.

For clarification purposes note that SAPs and CEPs are not just extensions to ports. The SAP, the CEP, and the port concept are rather specializations of the same base concept, which one could for example call "boundary interface concept". In contrast to the port concept, the SAP concept additionally has attributes (QoS, SAPI) and puts semantic constraints on the use of ports. The same is true for the CEP.

Having this new possibility at hand, we can now start identifying, comparing and analyzing architectural patterns of e.g. layer/plane compositions that are typically used. Figure 2 and figure 3 may stand for two of such typical patterns. This is subject to further research.

Acknowledgements: Many thanks to Lars von Wedel (LFPT, RWTH Aachen) for the discussion about CEPs. Furthermore, we would like to thank Andreas Witzel, Jörg Bruß and Dietmar Wenninger (all Ericsson) for their support.

References

[1] H. Balzert. *Lehrbuch der Software-Technik: Software Entwicklung*. Spektrum Akademischer Verlag, 1996.

[2] H. W. Barz. *Kommunikation und Computernetze: Konzepte, Protokolle und Standards*. Hanser, 1991.

[3] S. Boecking. *Object-Oriented Network Protocols*. Addison-Wesley, 2000.

[4] M. Broy. (Inter-)Action Refinement: The Easy Way. *Program Design Calculi, Series F: Computer and System Sciences*, 118, 1993.

[5] M. Broy. Compositional Refinement of Interactive Systems Modelled by Relations. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference*, LNCS 1536, pages 130–149. Springer, 1998.

[6] J. Eberspächer and H.-J. Vögel. *GSM – Switching, Services and Protocols*. Wiley, 1998.

[7] J. Fisher. Model-Based Systems Engineering: A New Paradigm. *INSIGHT – A publication of the International Council of Systems Engineering (INCOSE)*, 1(3), 1998.

[8] D. Garlan, J. Knapman, B. Møller-Pedersen, B. Selic, and T. Weigert. Modeling of Architectures with UML. In A. Evans, S. Kent, and B. Selic, editors, *≪UML≫ 2000 – The Unified Modeling Language: Advancing the Standard; Third International Conference, York, UK, October 2-6, 2000*, LNCS 1939. Springer, 2000.

[9] F. Halsall. *Data Communications, Computer Networks and Open Systems*. Electronic Systems Engineering Series. Addison-Wesley, 4th edition, 1996.

[10] D. Herzberg. UML-RT as a Candidate for Modeling Embedded Real-Time Systems in the Telecommunication Domain. In R. France and B. Rumpe, editors, *≪UML≫'99 – The Unified Modeling Language: Beyond the Standard; Second International Conference, Fort Collins, CO, USA, October 28–30, 1999*, LNCS 1723, pages 330–338. Springer, 1999.

[11] D. Herzberg and A. Marburger. The Use of Layers and Planes for Architectural Design of Communication Systems. accepted for publication and presentation at the 4th International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC 2001, Magdeburg, Germany; May, 2–4, 2001.

[12] D. Herzberg and A. Marburger. An Extended Model for Real-Time Systems. In B. Sanchez, N. Nada, A. Rashid,

T. Arndt, and M. Sanchez, editors, *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics (SCI) 2000*, volume 2, pages 200–205. International Institute of Informatics and Systemics, 2000.

[13] ISDN Protocol Reference Model. ITU-T Recommendation I.320, International Telecommunication Union, Nov. 1993.

[14] B-ISDN Protocol Reference Model and its Application. ITU-T Recommendation I.321, International Telecommunication Union, Apr. 1991.

[15] Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. ITU-T Recommendation X.200, International Telecommunication Union, July 1994.

[16] Unified Modeling Language Specification, Version 1.4. Technical Specification, Object Management Group (OMG), Feb. 2001.

[17] D. M. Piscitello and A. L. Chapin. *Open Systems Networking: TCP/IP and OSI*. Addison-Wesley, 1993.

[18] B. Selic, G. Gullekson, and P. T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, Inc., 1994.

[19] B. Selic and J. Rumbaugh. Die Verwendung der UML für die Modellierung komplexer Echtzeitsysteme. *OBJEKTspektrum*, pages 24–36, Juli/August 1998.

[20] B. Selic and J. Rumbaugh. Using UML for Modeling Complex Real-Time Systems. Whitepaper, Rational Software Corporation, Mar. 1998.

[21] A. S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 3rd edition, 1996.