Telektronikk 1.98

Information Systems Architecture

and telecommunication service platforms

Contents

Feature

Special

Status

Telektronikk

Volume 94 No. 1 - 1998
ISSN 0085-7130

Editor:
Ola Espvik
Tel. + 47 63 84 88 83

Status section editor:
Per Hjalmar Lehne
Tel. + 47 63 84 88 26

Editorial assistant:
Gunhild Luke
Tel. + 47 63 84 86 52

Telenor

Editorial office:
Telektronikk
Telenor AS, Telenor Research & Development
P.O. Box 83
N-2007 Kjeller, Norway
email: telektronikk@fou.telenor.no

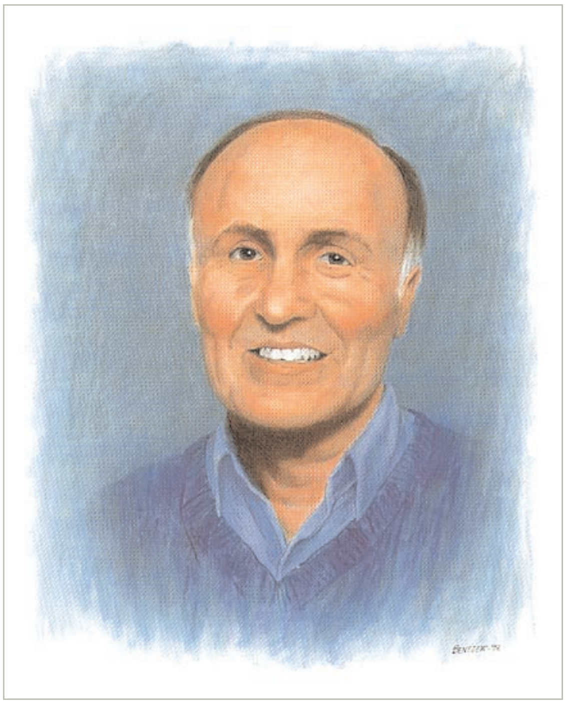
Editorial board:
Ole P Håkonsen, Senior Executive Vice President
Oddvar Hesjedal, Vice President, Research & Development
Bjørn Løken, Director

Graphic design:
Design Consult AS

Layout and illustrations:
Gunhild Luke, Britt Kjus, Åse Aardal
Telenor Research & Development

Guest editorial

ARVE MEISINGSET



When IT managers are asked in which area they have greatest need for contributions, they typically answer IT architecture. IT architecture is considered to be a means to control both evolution of systems and change of technology. A telecommunications operator typically manages several hundreds, maybe thousands of internal computer systems running on tens of different technologies. This picture is extended with software from different vendors in the telecommunications network itself, in service provisioning and for communication with vendors, partners, retailers and customers. The IT managers request means to manage current and future systems to serve the varying needs of the organisation in a timely manner. As telecom operators are replacing much of their personnel by computer systems, cost effective offering and management of these systems become critical to the survivability of the company.

The IT architecture subject has in some way or other been addressed since the first creation of computers. Systems planning problems have been addressed for a generation (30 years). Still, IT architecture at large is a poorly understood subject and some aspects have been shown relatively little attention by academia. One reason for this may be that large systems and large families of systems are only addressed by the industry and they exceed the size of academic studies. Also, the effects of a new systems plan may come ten years after introduction, when the original plan is forgotten and the analysts are gone. The design of individual smaller systems provides a more manageable and controllable size for theory development and validation. Added to this situation, the IT architecture subject is burdened with much sloppy terminology and marketing slogans. The fact that terminology and architectures are interwoven and motivated by marketing and power struggles within the organisation should come as no surprise to the critical reader. Therefore, the analysts will have to face many obstacles in the development of much needed rational approaches to IT architecture.

To make the scope of this issue clear, I have to introduce some terms which are frequently used with overlapping and undefined meanings. The term Information Technology, IT for short, covers a vast area of computer and telecommunications technologies, media, and their usage. I will use the term Information Systems, IS for short, to denote both manual and automatic systems used to provide or manage information to some users. Thus, an automatic IS denotes use of IT in a system which provides information to some users. Actually, an automatic IS manages data only. The data may provide information and/or knowledge to some users. When using the term IS, we consider the high level user related aspects of the systems and disregard the low level hardware related aspects. The term IT is misused in most organisations, as exemplified in the first two paragraphs of this introduction, as well. Organisations not producing the technology should use the term IS and avoid use of the term IT.

The term Information Systems Architecture, ISA for short, is used to denote the overall structuring of an information system or a family of interconnected information systems. We will decompose ISA into two orthogonal dimensions:

- An information systems plan provides an overview of what systems are planned to exist in some area and the interconnections between these systems
- A system architecture prescribes the structure of one system or a class of systems.

The term system appears in both these definitions. An information system we consider to be made up of a set of data provided with a mechanism to enforce the consistency of these data. An automatic information system can be centralised or distributed; however, it is the scope of the consistency enforcing mechanism which defines the boundary of the system and the boundaries between systems.

The assignment of functionality to systems we call evolution planning. Evolution planning is typically operational (one year range) or tactical (two – three years). More radical designs or redesigns of a family of systems, i.e. information systems planning, is considered to belong to strategic planning (typically three or more years into the future). This issue focuses on strategic rather than evolution planning. Also, we do not address in any detail the assignment of functionality to systems; neither the detailed design of user interfaces to systems. User interface design relates to systems planning, as data and interfaces are designed relative to tasks, and the tasks are defined relative to the data and interfaces they will manage. Hence, tasks and computer system boundaries have to be designed interactively, and not in a strict sequence!

We will use the term migration planning to denote the planning of migration from one system architecture and/or infrastructure to another.

The term system architecture denotes the structuring of the system IS itself, while infrastructure denotes the software and hardware environment in which the IS is running. The term infrastructure comprises the execution platform, as well as tools used to develop and manage the software. The term system software is used as a synonym to the term software infrastructure. The infrastructure may impose restrictions on the system architecture of the IS.

Frequently, evolution and migration planning are linked, as the functionality of a system is typically changed when the system is being migrated. This issue of *Telektronikk* addresses platform issues. Migration planning is not covered. However, needs for migration is motivating the papers on platforms. System architecture is addressed in some of the papers, but is not covered extensively. To separate the telecommunication network from service provision is one of the particular architecture issues for the telecom domain. Interoperation of alliance partners is another topic. Both these topics are addressed.

The previous paragraphs define not only the scope of this issue, but also the focus of current research on ISA in Telenor R&D. Furthermore, the paragraphs provide some indications of the limitations of this research, even if some planned papers are missing. The first paper of this issue provides an introduction to ISA. If information systems planning is paralleled with city planning, systems planning relates to which buildings should exist, and what communication is needed between these buildings. The platform issue relates to which basements these buildings should have. Even if these questions provide a limited view on ISA, they are fundamental questions of any architecture work, and I hope the reader will benefit from reading our questions and answers.

Arve Meisingset

Introduction to Information Systems Architecture

ARVE MEISINGSET

This paper parallels information systems architecture with city plans, building architecture and construction. The paper discusses the merits of systems planning and human-computer interface design at large, and discusses the dependence on organisation of users and tasks. Information systems architecture is as conflicting an area as city planning. The paper will highlight some of these conflicts. The information analyst may be provided with rationale techniques to analyse systems and organisations, however, for the choice of values and loyalty there is no objective and independent position. Also, information systems architecture is no unique field, but consists of several sub-fields, for which specific expertise and approaches are needed. The paper makes an attempt to distinguish these fields.

From solitude to abundance

A traditional Norwegian house – as we dream of it – is placed in nature, distant from other houses. It is a white house, accompanied with a red farmhouse, a fruit garden, a few flowers and small patches or strips of green fields, surrounded by a huge forest, mountains and fjords. This solitude can be compared with the early days of computing, when a few computer systems were surrounded by a mainly manual organisation. As a contrast to this romantic picture, Figure 1 provides a modern functionalistic view on architecture.

In the 1970s systems developers in Scandinavia [1] were pretty conscious of the social environment of computer systems, how systems were used and what co-operation between management and employees was needed for the success of their use. The project workers were conscious of what work should be automated and what should not. In particular, the critical school of systems development contributed to making the labour unions become constructive participants in the development of computer systems. This co-operation has been weakened in the 1990s, in particular as business process re-engineering (BPR) consultants [2] have brought the system-theoretical school with some new icons from USA back to Scandinavia and thought the news was progress. They do not seem to have knowledge of the historical development of Scandinavian systems thinking from the system-theoretical school (Langefors), via the socio-technical school (Høyer) to the critical school (Nygaard) and design of artefacts (Ehn) [3] (Dahlbom) [4], nor of their relationships to Taylor, Mumford, Marx, Hegel and Simon. Certainly not to activity theory [5] and Soviet philosophy [6] – or to John Dewey's *Theory of inquiry* (1938). These mentioned authors provide some of the basic texts for any inquiry into the design of manual or automatic information systems.

A multimillion dollar BPR project in Telenor has produced a stack a couple of metres high of PowerPoint paper copies and many promises in the internal press. Nothing of this is applicable for computer systems development – not even as surveying material, and neither may such usage have been intended. Our management claimed that the work was urgent and most important. However, the results from the project, if any, are highly questionable.

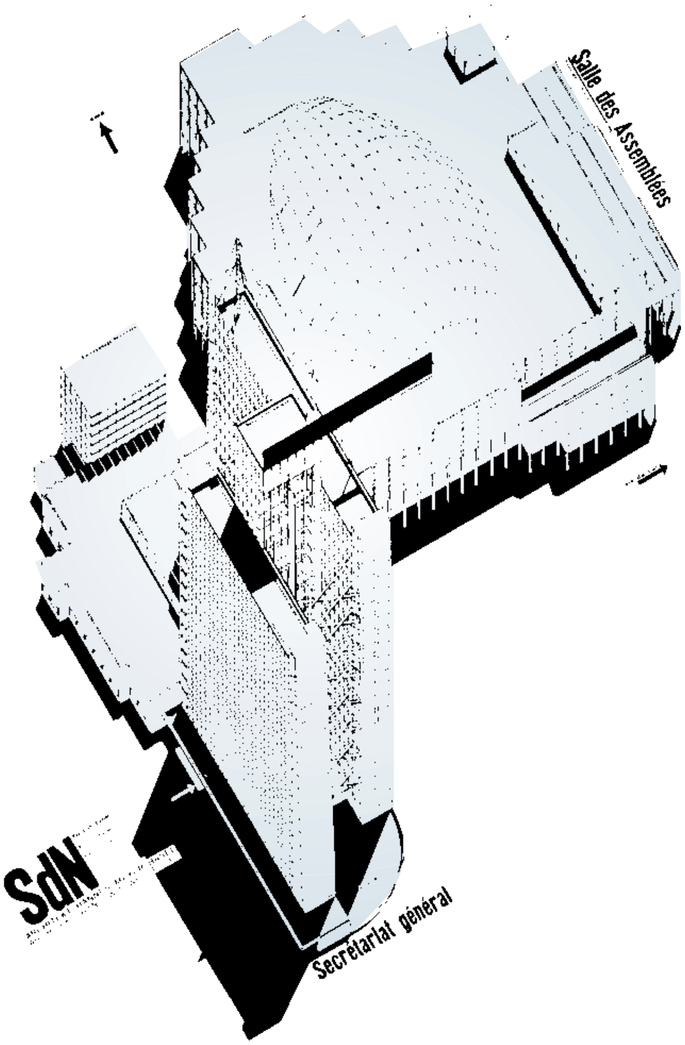


Figure 1 Hannes Meyer: Proposed design of the League of Nations' building in Geneva, 1927 (not built). In Hannes Meyer's opinion a completely rational building, where size, form and interior design are unconditionally determined by technical and functional requirements. The 'rationality' is underlined right down to the method of sketching and the choice of the print. (From Nygaard, E. Arkitektur i en forvirret tid : Internasjonale strømninger 1968–94. *Copenhagen, Christian Ejler, 1995.)*

Today, a large proportion of the office routines are automated by computers, and reorganisation of the company is just as much a question of organisation of computer systems as of organisation of people. The BPR project proves that many organisations are willing to spend much money on addressing computer systems in an organisational context.

Poverty and misconceptions

Much housing architecture in Norway is traditional, with old brown timber styles, white or otherwise painted houses from early this century, and fishing villages – all in wood. However, mass production, panorama windows, new materials, petrol stations, and bad functionalism have frequently provided a poor result. There are exceptions, like stave churches, stone churches, Hanseatic towns, old mining towns, a Jugend style town, advanced functionalism from this century and some good modern architecture of public buildings.

Many human-computer interface system designers seem to think that anything is good if they just use windowing, icons and pop-up menus. However, these are the architectural parallels to petrol stations and shopping centres. The graphics and contents take their models from books for pre-school children. The IS user interfaces are frequently of poor quality. If the programmers tried to publish on paper, their products – both content and editing – would frequently not be accepted.

Many current web-pages are cluttered with graphics – which increase both telecommunication transmission time and human interpretation time. Database applications frequently use panels with small windows for individual fields and tables, which clutters the overall relationships and presentation of the information. The metaphors for desktops have been used and misused in areas where they do not fit, for example in database management and process control. Widgets appropriate for the window frame are frequently and inappropriately put into the window working area. Windows are cluttered with information – like advertisements on a shopping centre window. This is the opposite end of the quality scale from what is provided by contentious and knowledgeable book printers. Therefore, we need old knowledge rather than new knowledge to human-computer interaction design. We need purity rather than overloading, and we need to investigate recurrent patterns other than current windows metaphors. Consciousness about aesthetic values and styles has to be radically improved. Fundamental questions about the design of user interfaces at large should be addressed: What are good user interface patterns of various kinds of computer systems? Is it really true that screen presentations should differ from paper presentations? Should we alternatively strive for compatibility between and independence of media?

Information systems planning parallelled with city planning

Norway has few examples of consciously planned architectures at large, where towns and landscapes are formed artificially, to support human living. However, the common Danish-Norwegian king Christian IV used quadratic street plans in his many renewed town designs, like Kristiansand and Oslo. There are a few examples of artificial landscape designs, like the Fredrikstad fortress and the Vigeland park. The most striking

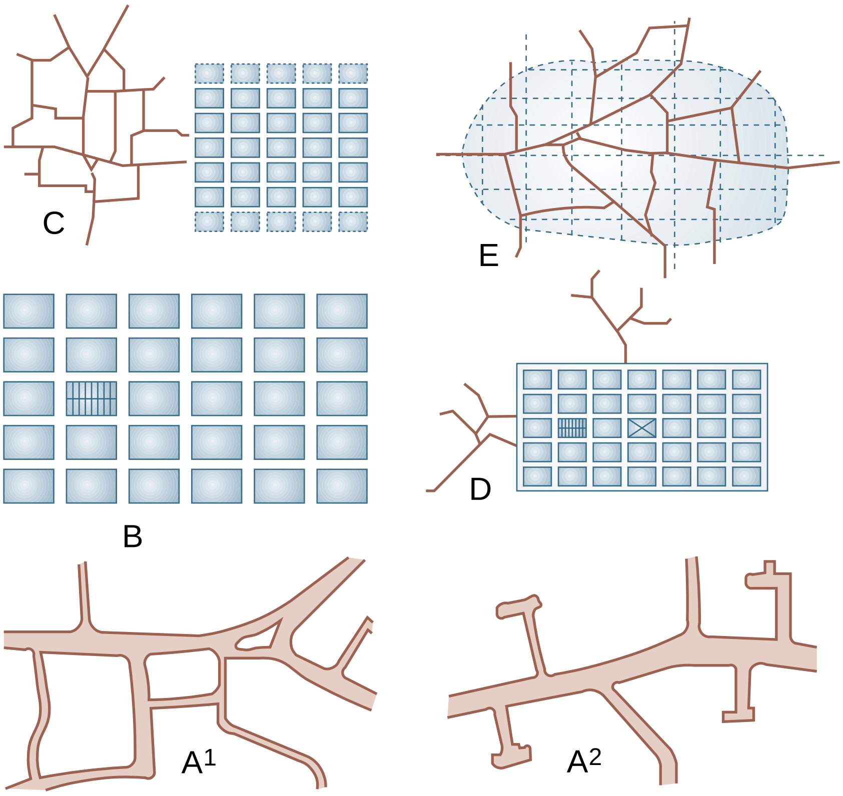


Figure 2 Organic growth and planned urban form diagrams.
Key: A – two characteristic kinds of Organic Growth: Western European (A^1), providing for street frontage plot development and Mesopotamian/Islamic (A^2) with housing access culs-de-sac; B – the gridiron as the usual basis of Planned Urban Form; C – an organic growth nucleus with planned gridiron extension, loosely based on Edinburgh; D – a planned gridiron nucleus with organic growth extension, loosely based on Timgad; E – the special three-dimensional Western European circumstances whereby an early medieval organic growth pattern was superimposed on the abandoned gridiron of a temporarily deserted Roman city – based on Cirencester, England. (From Morris, E A J. History of Urban Form : Before the industrial revolutions. *Longman, Scientific & Technical, Third edition, 1994.)*

examples, though, are roads, railways and bridges that have been transforming the landscape of modern Norway. Bridges and roundabouts have become objects for artistic expression. Figure 2 provides some conceptions of city plans.

Information systems planning is in [7] compared with city planning. Information systems planning is as different from the development of individual computer systems as city planning is different from building design.

City planning – or the lack of it – has profound impact. Motorway interchanges, brick, asphalt, aluminium, glass, cars, noise and exhaust can have a devastating impact on the living conditions of the inhabitants, while organic German villages or harmonious St Petersburg provide much richer conditions. Villages are frequently not strictly planned, but regulated by some kind of consensus, while St Petersburg is developed according to a grandiose plan. The examples show the importance of choosing an appropriate perspective on city planning and of realising what interests are being served. The unlimited power of Peter the Great may help when developing a city plan, but more modest approaches may provide just as good results.

The current situation in a company like Telenor is that most data and functions are already automated. Some computer systems are up to 20 years old, use old technology, are adapted to an organisation extinct many years ago, and are not supporting new technology in the telecommunication network. Neither may they support new services to the customer, new organisation of work, and service management by the customers themselves. Examples of such mis-adaptations exist, even if this is not a balanced presentation of IS in Telenor. However, these are some of the reasons why IS architecture is listed as the highest priority work item by most IS managers [8] [9]. The prescription used for this disease is frequently to provide a business information framework [10] [11] [12] [13] to survey all systems, data and work procedures of the entire organisation and to implement procedures for requirement capture and implementation. These approaches are at great risk of controlling everything, but creating nothing. A focused approach on a problem area which needs attention can be much more successful than attempts at developing a grandiose plan for everything. The preferred degree of integration can be different for components of one system, for systems within one business unit, for business units within one corporation, for partners within an alliance, for vendor-customer relationships and for competing parties requiring some co-operation. Each kind of co-operation may need different focus and approaches to systems and communication planning. Note that still other techniques may be needed to define software components and identify commonalities and differences to support reuse and sales of software components to customers having similar but different needs.

Box 1 Terms and definitions

Systems planning	- identification and delimitation of what systems should exist, and identification of the relationships between systems
Communication planning	- identification and standardisation of communication flow, contents and means
Evolution planning	- planning of transition from current systems to planned systems
Systems framework	- the analysis and design environment provided to the developer, within which he undertakes his work
System architecture	- structuring of each system as seen by the system developer as the primary user of the architecture
System development	- surveying, analysis, design, implementation, testing and installation of an individual system or a component of a system
Change management	- management of change requests, analysis of change effects, management of changes and provision of the changed system
Infrastructure	- the total environment in which a system is running and supported
Platform	- the hardware, software and middleware on which the system is running
Migration planning	- planning of transition from current infrastructure to new infrastructure
Software portfolio planning	- planning of software components to be provided to some market, and which can be variously configured in customer systems

Organisation struggle

Frequently, IS architecture is by the IS managers considered as a means to empower the IS department itself. By introducing strict management procedures, the IS department hopes to acquire control of the situation and not to be dependent on the unpredictable will of the business units only. The IS department has often been frustrated by the business units introducing new technology in the network and for service provision, and only thereafter ask for administrative support by computer systems. Therefore, a co-ordinated business, technology and computing strategy is sought for. Also, a corporation wide IS department can claim to have a more global perspective than the individual business units. However, the business units are created to provide more individual freedom and flexibility than what can be achieved by one centralised organisation. The business units are established in a period of great change in technology, services, markets, regulations and competition. The divergence of the corporation into business units can provide the corporation with dynamic powers to compete with small independent companies [14]. This indicates that centralisation of the IS department and the split of the corporation into several business units may serve conflicting interests. The conflicting directions can also be interpreted as attempts by the top management to 'ride two horses' into an uncertain future, or it shows lack of a clear strategy. The IS analysts will in this situation be forced to choose between conflicting interests, as described by the critical school of systems development – there is no independent and objective position. However, the analyst can investigate consistency and consequence of choices, and he may try to balance conflicting interests. The identification and formulation of strategies and perspectives are central themes and means of power struggle.

Current IS architecture work has replaced much of the previous 'data modelling' work. During the 1980s several organisations attempted to define and harmonise use of terms throughout the entire corporation. Some of this work proved very useful, like introduction of organisation wide customer identifiers, product identifiers, etc. However, most of the work lacked focus and priorities. New data definitions were developed for systems to be implemented 5–10 years into the future, and proved to be non-appropriate or forgotten when needed. A more focused approach to defining data for emerging systems only, and to co-ordinate data only when found beneficial and urgent could have produced better cost-benefit. The data administration of the 1980s centralised power and bureaucracy within the IS departments without producing the prospected benefits. Decentralisation of data administration to individual business units with co-ordination between business units only when needed could have produced better results and a more sustainable organisation of the work. The central data administrator in the IS department could then have concentrated on initiation, teaching, supervision and co-ordination, without being responsible for managing his own data administration people and their tasks.

During the 1990s, IS architecture has received much of the attention given to data administration during the 1980s. The challenges being addressed are greater than those facing the designer of individual systems. The reasons for this are the greater size of the problem of analysing the whole corporation rather than an individual system, problems with choice of an appropriate level of detail, abstraction versus the concrete, organisation dependence versus independence, technology dependence versus independence, and short term versus long term planning. The risks for misconceptions, unfocused organisation of work, and use of inappropriate techniques are very high. Only a small portion of the efforts put into IS architecture will produce useful results. Also, little or no appropriate education in – and knowledge of – the field is available. A manager may be surprised by this situation; however, if he compares with city planning, he should not be surprised. Despite the risks, misadaptions between computer systems and to the organisation of the company make IS architecture work most needed.

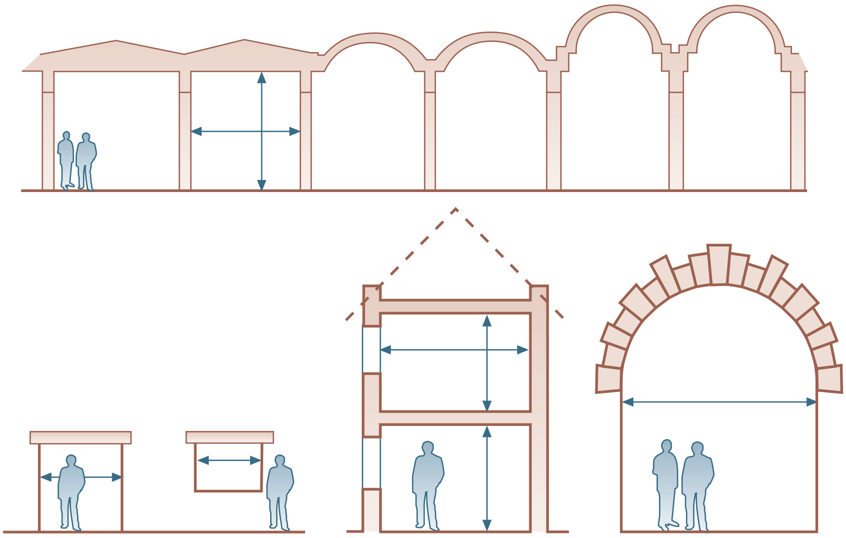


Figure 3 Construction materials: characteristic room sizes, height of building and size of openings in walls, as constrained by traditional 'local vernacular' materials and technology, related to the scale of the human figure.
(From Morris, E A J. History of Urban Form : Before the industrial revolutions. *Longman, Scientific & Technical, Third edition, 1994.)*

Information system architecture paralleled with building design

A traditional Norwegian home applies a functional layering of its floors: The cellar is used as washroom and as food store for potatoes, cabbage, canned food, etc. The ground floor contains the entrance, living rooms and the kitchen. The bedrooms and a bathroom are found on the first floor. The loft is used to store away unused furniture, carpets and maybe dried meet and fruit. Figure 3 shows some architectural patterns.

Similar functional layering software architectures are being developed for computer systems; however, their role and purpose are more unclear. Software architecture is concerned with the organisation of software systems, functions and data as perceived by the end users and developers.

The client-server architecture arrived with personal computers and workstations allowing more computing to take place in the client. Fat clients have been found easy to implement, but thin servers are not found very efficient for serving many users simultaneously. Fat servers can be efficient, but provide in principle not much different from IBM 3270 alphanumeric and graphical dialogues on dumb terminals available in the early 1970s. Today, this is called network computing. More balanced architectures have most often been inefficient to implement, because functions have to be carefully split between the nodes, and the interaction between the nodes is frequently command-based. The three-tier-architecture provides a principal split between external presentation, application and internal storage of data. However, it has the same difficulties with providing efficient development as with balanced client-server architectures. The reason for the difficulty may be that they do not distinguish between specification and implementation. This is accomplished by the three-schema architecture [15]; while the specifications are split into external schemata, application schema, and internal schemata, the execution architectures can be generated in various ways automatically. Unfortunately, not many developers and tool vendors know the three-schema architecture. The three-schema architecture realises that software can be nested [16], becoming more abstract than housebuilding. Both usage and development/compilation dimensions of software can be realised by similar or identical transformations, and this way, the software can be put in series and parallel [17], as well.

The subject software architecture is in [18] split into four categories: (i) architectural description languages, (ii) codification of architectural expertise, (iii) frameworks for specific domains, and (iv) formal underpinnings for architecture. The book uses the specification language Z, based on predicate calculus, for the first category. As exemplified in the Urd method – see next paper – the difficulty is not only what language should be used, but what notions should be described in an architecture. The book describes the following architectural styles of the second category: pipes and filters, data abstraction and object-oriented organisation, event-based and implicit invocation, layered systems, interpreters, process control, distributed processes, subroutine organisation, state transitions, reference architectures for specific domains, and heterogeneous architectures. Within the second category, the book provides a design space for user-interface architectures. Other papers of this issue of *Telektronikk* provide contributions to the third category, on the data-oriented, distributed and telecom sub-domains. The fourth category provides means to reason on architectures.

Frameworks

I believe building architects can specialise in different kinds of buildings, and they can have menus for what kinds of features have to be provided for these buildings. These menus are not prototype designs of the buildings, but they are frameworks for various designs.

While a software architecture prescribes the structure of the final systems, a framework prescribes the structure of the specifications of these systems, functions and data. We use the term management framework when this structure aims at describing and controlling the whole corporation and not just the structure of the existing and planned software systems, functions and data. The management framework aims at providing the IS architects with control of their total environment.

For software systems there are frameworks, reference models, architectures or prototype designs which apply to specific application areas, like user-interface software or database applications. Also, there are frameworks for distributed databases, distributed processing, long transactions, telecommunication services, telecommunication management, telecommunication networks, and object orientation. For some areas there are competing frameworks, and some are overlapping. Some areas are specific, and some are believed to be very generic.

Open Distributed Processing, ODP, [19] provides a framework for specifying and implementing communicating systems. However, the overall architecture of the final system is provided in the computational viewpoint of the ODP specifications. Hence, ODP is called a framework and not an architecture; however, the framework can be considered to be a meta-architecture of the final systems, functions and data. The three-schema architecture applies the same three layers both in the architecture and in its meta-architecture. ODP does not contain any layer or viewpoint similar to the external layer of the three-schema architecture, nor does the pure three-schema architecture contain any layer for partitioning and communication. Hence, the two architectures/frameworks overlap, but do not completely cover each other.

Some frameworks can be used in several different ways. The ODP reference model can be understood as follows: the enterprise viewpoint contains all business policies as well as requirements on the systems of the studied application domain; the information viewpoint contains all data definitions, their behaviour – and grouping into systems, screens and reports; the computational viewpoint contains the vertical partitions of these definitions; the engineering viewpoint contains the horizontal configuration of the systems; while the technology viewpoint contains the implementation designs. Each viewpoint is a complete specification of the entire application domain from one perspective; only the technology viewpoint is needed to implement the systems.

[20] provides a different interpretation of use of the ODP reference model: the enterprise viewpoint contains an identification of functions or tasks to be supported by the systems; the information viewpoint contains data definitions and constraints on the behaviour, but not functional derivations; while the computational viewpoint contains the processing prescriptions and interfaces between processes; the engineering viewpoint defines the functions needed to support distribution; while the technology viewpoint identifies the technology needed to implement the systems. Here all viewpoints are needed to provide a complete specification, and references are made between viewpoints to provide completeness.

The two interpretations of the ODP reference model may produce totally different specifications and implementations.

James Martin provides a management framework in his book [10] together with prescriptions for development processes and methods. The Telenor Business Information Architecture, BIA, [11] and the BT Tile Architecture [12] primarily provide management frameworks for business processes, tasks, data definitions, functions and systems, and not elaborate methods. In particular, BIA has a strong focus on business goals. On the other hand, the Urd method takes business plans and overall organisation of the corporation for granted, and analyses software systems and their organisation only. Urd provides no management framework.

Frameworks can be specified as viewpoints, roles, function blocks, data schemata, data flow, control flow, precedence relationships, predicates, objects or other. Even if some initial comparisons of frameworks – and architectures – are provided in this issue of *Telektronikk*, we have rather shallow knowledge of how to specify frameworks, what are the (hidden) implications of use of different specification techniques, what scope of validity is provided by each framework, which benefits and drawbacks do they provide, and how are the frameworks compared and evaluated. These questions may not only be of academic interest, but can have huge practical implications on how planning and development work is pursued – and on the final designs.

Methods

A method should define the goal for a piece of work, and a way of undertaking the work to achieve the goal. Both focus on the goal and conscious control that the work leads to the goal are important to produce effective results.

In the mid-1970s, I participated in using the ISAC [21] method to analyse a large portion of the information handling in Telenor. We made decompositions on up to seventeen levels, some nodes in some graphs were trivial, others were very complex, without us knowing this. We, as analysts, felt that we lost control of the abstract decomposition, and we had just vague ideas about how to use the analysis results in the subsequent design. Much work on large systems is like this, you can feel lost, and then some magic happens – provided by a key designer, who takes responsibility and provides what is needed irrespective of the theory predicated.

A specification of automatic information systems is comparable to a scientific theory in the way it corresponds to people outside the automatic system, how it describes the managed application area, and how it prescribes the structure and behaviour of the automatic information system itself. See a separate paper on these three dimensions in this issue of *Telektronikk.* A method for developing and validating the specifications of automatic information systems is comparable to a philosophy of science, and much could be learned, but is not yet done, by comparing these.

The ISAC method [21] is primarily a method for the development of individual systems, but addresses some aspects of systems planning, as well. The Urd method provides an approach to systems planning only, and no approach to analysis and design of individual systems.

An approach to systems planning and development can be evolutionary or revolutionary. The analysts have to consider carefully what is the most appropriate time perspective and approach. The choice will depend on the situation: market needs, user requests, reorganisation of work, change of technology, change of communication interfaces to other systems, and change of data definitions and entities of the application domain. The Urd method provides both a corrective and an idealistic design phase. Even if evolutionary design is not the purpose of this approach, it provides ‘evolutionary’ intermediate results on the road to developing a revolutionary new design for the application area.

Process organisation

The approach to design a house is dependent on what tools and prefabricated elements are available. So also for information systems.

A method is a way of thinking, while a process prescribes how work may be organised into separate activities, i.e. phases, steps or other, to achieve the goal. Activities should provide some visible useful results to somebody, and should be delimited by some reporting and decision. The process structure is influenced by the method used, but steps in the method may not be identical to steps in the process, e.g. you may do surveying, analysis and some design in one and the same step of the process. [22] provides an overview of state-of-the-art of process definition (research).

The Urd method primarily prescribes organisation of activities, i.e. process organisation, but contains method statements, as well. The Urd subject analysis phase contains both analysis of data and design of subjects in one and the same phase.

Engineering

Building construction involves a lot of engineering and handicraft professions in areas like insulation, dampproofing, electricity supply, water supply, carpentry, plumbing, light planning, heating, locksmithing, painting, colour planning, interior planning, decoration, etc. Larger buildings require many more professions, like statics, material sciences, fire planning, ventilation, automation, accommodation, car parking, industrial architecture, hotel architecture, supermarket architecture, etc.

Most contributions from software people are really on engineering issues and not on the more high level architecture issues. Therefore, implementation and architecture issues are frequently confused. The implementation issues can comprise choice of communication protocols, database management tools, repositories, screen definition languages, programming language (styles), etc.

Portfolio planning and globalisation

In times gone by, building elements were typically hand-crafted on location. Today, new building elements are typically produced in specialised factories, collected into modules by module producers, and installed by a professional team in accordance with a standard architecture.

Software vendors will typically sell their software to many organisations. The IS department will typically want to reuse their objects, modules, systems and infrastructure within several user organisation units. The reuse can be cost-effective for vendors and customers alike. However, there are a lot of concerns to be made: The software should fit into an existing organisation, business, culture and language, and it should co-operate with existing objects, modules and systems, which most often are different for different organisations. How to identify areas for reuse, how to define elements suited to reuse, how to provide infrastructures for reuse, and how to incorporate reuse considerations into methods and tools are important questions. Use of one (kind of) system to support several business units is not and should not be the normal answer to these questions. Portfolio planning is an important topic, concerned with the product planning, co-ordination and marketing from the development organisation. Portfolio planning should not be confused with systems planning of the customer organisations, e.g. in the business units of the corporation. However, it is the co-ordination of portfolio and systems plans that can provide the benefits to the corporation.

The enterprise is not the only possible scope of portfolio and systems planning. The aim of component technology is to provide software components that can be reused in various environments. The entire globe can be the marketplace [23]. Effective use of component software raises a set of fundamental questions: Will a designer first have to write a detailed specification before he can search for a component, which he can use? If the implementation is generated automatically from the specification, what is then the benefit? How should a component be prepared for specialisation and extensions? How will a component depend on other components and their environment, and how will the designer learn about this, and be guided in his designs? Will the designer operate as a librarian, rather than as a creative designer? Will there be reuse of components within certain schools or application areas, in which the designers get specialised training, and not much across these schools? The standardisation of the telecommunication management framework, TMN, within the International Telecommunication Union, ITU, may define such a school. Future TMN software may be based on downloading TMN object definitions from the ITU web.

Design implications

Systems planning may have as good or devastating impact as city planning, and the implications of alternative designs are at least as poorly understood.

Information systems planners need to choose design perspective and serve the interests of different stakeholders just as consciously as any city planner. However, you may know a good city plan when you see it, but it is hard to tell what characteristics make a good city plan. Computer systems planners need to acquire knowledge of what is a good systems plan. The Urd method aims at identifying systems in which consistency of data has to be enforced, and which serve the manual organisation in the best way.

Some approaches to systems planning [11] claim to ‘provide the glue to bring together any fragmented organisation’. However, this may neither be an appropriate goal, nor may the approach provide techniques to analyse similarities and dissimilarities between organisation units. Frequently, they do the opposite, they decompose tasks and data in such a way that the particularities are not revealed, and they provide no means of comparison, generalisation and adjustment. No surprise, this lack of constructiveness is called ‘logical’.

In the Urd method great care is taken to survey and analyse the concrete aspects of the organisation, its routines, data and systems, without going into a too deep level of detail. Also, a concrete systems plan, with reference points between systems, is provided as the end result. Data (relationships), with their subordinate behaviour (called methods in object oriented languages), are grouped into systems.

The approach and perspective taken in the Urd method are distinctively different from that of approaches which identify functions of the application domain [10] [11] [12] [21]. Frequently the function oriented approaches do not distinguish between manual and automated tasks, and they do not provide rules for what should be considered one or two functions – in sequence, or in parallel. The final outcome of using the approach may be a set of functions and groups of data. These functions and groups may in no direct way relate to systems and their interconnections. Hence, separate mappings from these notions to design notions are needed. As mentioned, the results of applying the Urd approach and some other approaches can be very different. This proves that the analysts must be conscious about what perspective and which approach to choose for their systems planning.

The Urd method may be appropriate for co-ordinated design of a family of systems. However, Urd is inappropriate for design of communication solutions between more independent organisations and systems. For this purpose, a broker kind of solution will be more appropriate. This issue of *Telektronikk* provides two papers on this topic, but provides no method for planning and co-ordinating such exchange of information. The analysis method needed may be dependent on the solution envisaged.

Change and use

It is being claimed that the most permanent entity in Oslo is the continuing construction work. That the construction work is everlasting may be true. However, while each construction project lasts typically for some months, each house may last for tens or hundreds of years.

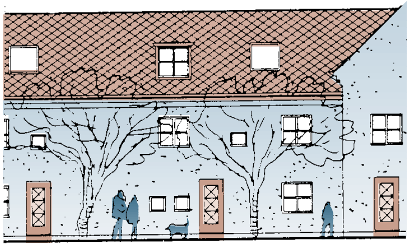
Large automated information systems may have a typical lifetime of twenty years. There is no indication that this typical lifetime has changed significantly over the history of computing or that it will change in the near future. However, the time periods between new versions may change depending on software environments, tools, developer organisation, and end user organisation wish and capability to adopt the new versions.

There has been a lot of talking, thinking and implementation of modularisation of houses. However, the effects at large are hard to spot. It is typically small entities like doors, windows, mirrors, toilets, wallpaper, etc. which are reused, and their effects are enormous.

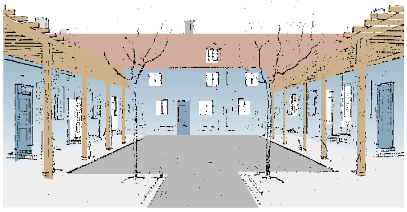
Notions like modularisation, reuse, object-orientation, etc. have been very popular in programming. Certainly, a ‘divide and conquer’ strategy has been needed to implement any large system, including use of software developed by others. Reuse of application independent software, and of application dependent data types, is certainly used. However, the frequency and benefits of real reuse of application dependent code is uncertain, and it is more uncertain what ‘preparation for unpredictable change’ can mean. What does reuse mean? Can centralisation of code do much the same as inheritance? Can copying of code do much the same? Is reuse within and between applications not so much of interest? Will the real benefits come from standardisation of data types and objects for entire application areas, and downloading and configuration of these more or less ready-made applications?

Some schools [24] advocate adoption of generic software, which is specialised by the users themselves. Web-sites and drawing packages are examples of this. Web-sites can become large for a large user group, and hence, personalised tailoring should be discouraged. Drawing applications may be prepared for smaller user groups, and invites tailoring.

A person may design and build a small cottage without an architect, and he may decorate and adapt the room to his usage. However, when it comes to larger and maybe more long-lasting constructions, architects and engineers are needed.



Boje Lundgaard and Lene Tranbjerg: Housing units at Blangstedgård, 1988



Heinrich Tessenow: Town house project, 1911

Figure 4 From Nygaard, E. Arkitektur i en forvirret tid : Internasjonale strømninger 1968–94. *København, Christian Ejler, 1995.*

We believe that the situation is similar for information systems. Systems planning and system development for large systems are separate professions requiring a large set of surveying, analysis and design techniques in addition to knowledge about technology, organisation and specific application areas. New technology, like webs and other generic software, may change the borderlines between what requires high expertise and what can be handled by more novice developers or the end users themselves, but the need for the information systems architects and engineers remains.

A particular problem for large organisations is the cascading of changes. Changes may cascade within one system due to a poor structure of the system. More challenging is the cascading of changes between systems. Here, detailed repositories are needed to trace effects of changes via several systems. Change management systems are needed to coordinate the changes, and responsibilities and costs of changes have to be shared.

Maintenance

Every owner of a house knows the costs of maintenance.

So does the owner of an information system. The speculations on reuse have frequently been motivated by hopes or intentions to reduce maintenance costs. However, there are reports [25] telling that adaptive maintenance can reduce the maintainability of the program code. Therefore, after some time, replacement of a family of systems can provide benefits which cannot be obtained by adaptive maintenance. Finally, replacement of systems can be cheaper than the initial development of these systems. This is an additional reason for pursuing long range strategic and idealistic planning of information systems, and not just short to medium term adaptive maintenance. The planner should note, however, that the new systems should not just replace the old ones, but provide new features and utilise opportunities not addressed by the old systems.

Aesthetics

Building architecture is not only concerned with functionality, but is concerned with aesthetic values, as well. Figure 4 provides some example building designs which are believed to be nice.

The aesthetic aspects of information systems architecture are in particular poorly understood. Should aesthetic aspects of a systems plan or a system design have any significance? What does it mean? Does it mean simplicity, understandability, symmetry, richness, or whatever? If object-orientation is considered good, is then the architecture good as long as it uses object-orientation? Is it our beliefs about the values that matter, or are there some more fundamental properties of the architecture itself that makes it nice or good? These are some of the practical and philosophical questions which should be addressed concerning the aesthetic aspects of information systems.

References

1 Bansler J. Systems Development in Scandinavia : Three theoretical schools. *Scandinavian Journal of Information Systems,* 1, August 1989.

2 Hammer M, Champy, J. *Reengineering the corporation : a manifesto for business revolution.* London, Nicholas Brearly, 1993.

3 Ehn, P. The Art and Science of Designing Computer Artifacts. *Scandinavian Journal of Information Systems,* 1, August 1989.

4 Dahlbom, B, Mathiassen L. *Computers in Context : The Philosophy and Practice of Systems Design.* Cambridge, Blackwell, 1993.

5 Bødker, S. A Human Activity Approach to User Interfaces. *Human-Computer Interactions,* 4 (4), 1989.

6 Bakhurst, D. *Consciousness and Revolution in Soviet Philosophy : From the Bolsheviks to Evald Ilyenkov.* Cambridge, Cambridge University Press, 1991. ISBN 0-521-40710-9.

7 Veryard, R. *Information coordination : the management of information models, systems and organizations.* Englewood Cliffs, NJ, Prentice Hall, 1994. ISBN 0-130099243-7.

8 Lederer, A L, Salmela, H. Toward a theory of strategic information systems planning. *Journal of Strategic Information systems,* 5 (3), 1996.

9 Gottschalk, P. A Review of Literature on Implementation of Strategic Information Systems. *Norsk Informatikk-Konferanse, NIK.* Trondheim, Tapir, 1996. ISBN 82-519-1381-0.

10 Martin J, Leben J. *Strategic Information Planning Methodologies.* Englewood Cliffs, NJ, Prentice Hall, 1989. ISBN 0-89435-358-6.

11 Telenor IT. *Business Information Architecture.* Oslo, unpublished.

12 Furley, N. The BT Operational Support Systems Architecture. *BT Technology Journal,* 15 (1), 1997.

13 Modelware International. *Introduction to the IFW A/B Level Data Model.* 1998.

14 Katz, H (ed). *Telecommunications : restructuring work and employment relations worldwide.* Itacha, ILP Press, 1997.

15 Griethuysen, J J van (ed). *Concepts and Terminology of the Conceptual Schema and the Information Base.* Geneva, ISO, 1982. (ISO TC97/SC5 N692.)

16 ITU-T. *Data-Oriented Human-Computer Interface Specification Technique : scope, approach and reference model.* Geneva, ITU, 1993. (ITU-T Recommendation Z.352.)

17 Meisingset, A. A Data Flow Approach to Interoperability. *Telektronikk,* 89 (2/3), 52–59, 1993.

18 Shaw, M, Garlan, D. *Software Architecture.* Englewood Cliffs, NJ, Prentice Hall, 1996. ISBN 0-13-182957-2.

19 ITU-T. *Reference Model of Open Distributed Processing. Draft.* Geneva, ITU, 1995. (Recommendation X.901.)

20 ITU-T. *Application of the RM-ODP framework.* Geneva, ITU, 1996. (ITU-T Recommendation G.851.1.)

21 Langefors, B. *Theoretical Analysis of Information Systems.* Oslo, Universitetsforlaget, 1966.

22 Conradi, R, Chunnian, L. Revised PMLs and PSEEs for Industrial SPI : Object-Oriented Technology. In: *Object-oriented technology : ECOOP/97 workshop reader, Jyvaskylä.* J Bosch, S Mitchell (eds.). (Lecture Notes in Computer Science; 1357.) Berlin, Springer, 1997.

23 Murer, T. The Challenge of the Global Software Process : Object-Oriented Technology. In: *Object-oriented technology : ECOOP/97 workshop reader, Jyvaskylä.* J Bosch, S Mitchell (eds.). (Lecture Notes in Computer Science; 1357.) Berlin, Springer, 1997.

24 Mørch, A. Evolving a Generic Application into a Domain-oriented Design Environment. *Scandinavian Journal of Information Systems,* 8 (2).

25 Kaasbøl, J J. How evolution of information systems may fail : many improvements adding up to negative effects. *European Journal of Information Systems,* 6 (3), 1997.

Arve Meisingset is Senior Research Scientist at Telenor R&D. He is currently working on information systems planning, formal aspects of human-computer interfaces and middleware standardisation. He is ITU-T SG10 Vice Chairman and the Telenor ITU-T technical co-ordinator.

e-mail:
arve.meisingset@fou.telenor.no

The Urd Systems Planning Method

ARVE MEISINGSET

This paper summarises a method for the identification and delimitation of computer systems within one business unit. The method defines what systems should exist within an application area, defines the subject contents of these systems and the boundaries between the systems. A more detailed exposition of the method is found in [1].

1 Systems planning

Systems planning comprises specification and initiation of development of a family of systems, while system development comprises specification and development of individual systems only.

Our approach to systems planning is abstract in the sense that it identifies subjects (being aggregates of data classes), activities and systems at a high level, without addressing the detailed design of data, functions and use of the individual systems.

Our systems planning approach comprises the identification of:

- systems
- subject contents of systems
- reference points between systems
- applications of systems
- co-operation between applications
- manual activities
- routines and data flow between activities and systems
- opportunities, costs and impact
- priorities.

Our approach is constrained by premises given by:

- business planning; choice of business areas, markets and market strategies
- high level organisation; future split into business units and degree of co-ordination.

Systems planning addresses the use of computer systems, and does not address

- IT architecture; principles, technical solutions and organisation of IT support
- role of development projects; identification and prioritisation
- role of maintenance work; identification and prioritisation.

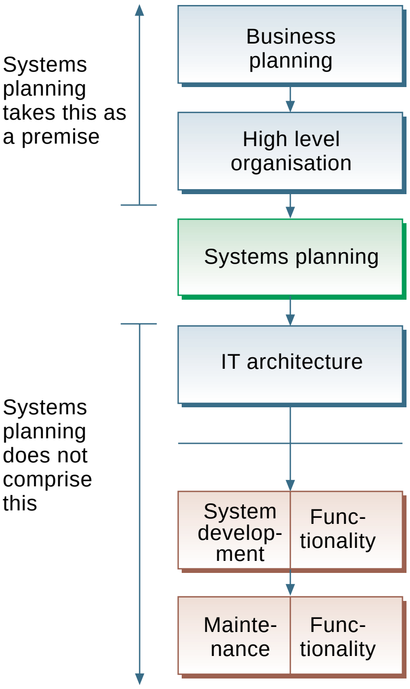


Figure 1.1 The main planning stream

Our approach has a main stream, depicted in Figure 1.1, but feedback loops must be observed. As an example, opportunities and constraints of the computer systems may provide ideas for change of the business plan and of the high level organisation.

Our systems planning method is developed to facilitate internal systems planning

- in one business unit
- for business units which already have several computer systems
- for strongly data oriented applications.

The method does not address

- co-ordination between loosely coupled business units
- systems planning for new business units/functions
- highly mathematical or process oriented applications.

Our systems planning method is constrained by the use of decentralisation as a political means of splitting the enterprise into several business units, but allows centralisation of systems and activities as a means to improve efficiency within a business unit. However, systems can be partitioned into (co-operating) applications as a means of decentralisation within that business unit.

The method takes

- existing systems
- existing organisation of work
- problems and opportunities

as its starting points.

The method delivers applicable intermediate results by

- identification of problems and opportunities
- overlap between existing systems
- a corrective systems plan
- an idealistic systems plan.

We believe systems planning should take surveying of existing knowledge of systems, applications and activities as the starting point. Our analysis method focuses on the identification of subjects which in principle can form a data base each.

Our idealistic design method groups subjects into systems to support the most efficient design of activities and routines.

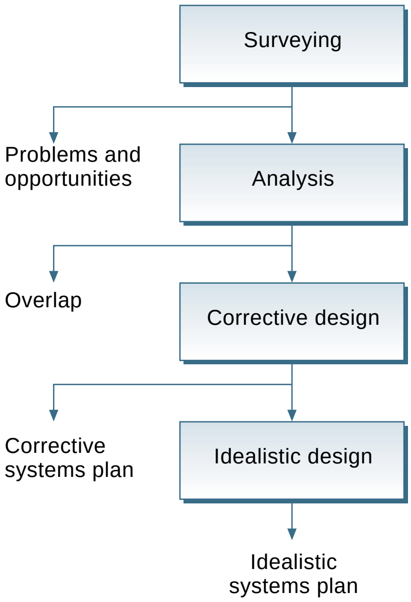


Figure 1.2 Phases in systems planning

The analysis work in all phases is a top-down decomposition of relations and dependencies. This knowledge is used in a bottom-up design.

The method is not abstract in the sense of being independent of organisation, routines and data design. Our method relates to concrete organisations, routines and data designs.

We do not believe in

- the existence of a universal method for all systems planning
- an all-comprehensive planning of all systems of an enterprise
- simultaneous application of a phase/step to all systems.

We believe planners should choose techniques and application areas according to specific needs:

- the planning method should be adjusted to the specific problems of the application area
- the planning should focus on the systems and applications considered to be of greatest importance
- use of phases/steps to different application areas need not be synchronised.

Systems plans frequently end up as ‘paper results’ without sufficient focus and determination on implementation. Therefore, the systems plan should be anchored by the customer, who must be engaged in prioritisation and start-up of implementation projects.

Compared to alternative approaches, our method provides

- less detailed analysis than [2]
- more detailed analysis than [3]
- top-down surveying followed by bottom-up design, as opposed to top-down-only approaches, like [3]
- identification and delimitation of systems, as opposed to identification of functionality only, like [3], [4].

We do not think that a cook book for systems planning can provide a unique guide to obtaining a good plan. In order to obtain successful results, a good cook is needed as well, and the planners have to choose elements from the cook book which are relevant to the needs of their application domain. Choice of the appropriate level of detail for the planning is

Box 1 – Terminology

1 System

A system is a computer system which can enforce the integrity of its own data. A system is in the systems planning defined to be a collection of data types (aggregated into subjects); therefore, we do not consider properties of the system related to external user functionality or internal storage and accessing of data.

2 Application

An application is an instantiation of (a vertical partition of) a system together with a (horizontal and/or vertical partition of) a set of data instances. Applications can be independent, loosely coupled or integrated, i.e. strongly coupled within the scope of the system.

3 Subject

A subject is an aggregation of data types which in principle can make up an autonomous system. A subject is defined to contain a set of references between object classes, with associated object classes and name bindings.

4 Reference point

A reference point indicates communication needs/redundancy between subjects or systems. An object class which belongs to several subjects or systems is called a reference point. A reference point can be n-ary.

5 Name binding

A name binding prescribes that instances of the subordinate object class are identified locally to an instance of their superior object class by their relative distinguished name. Name bindings used in the systems planning can form trees of object classes only, never networks.

6 Reference

A reference is a pointer attribute from one object class to one other object class. Pointers can be one-way or two-way, i.e. two mutually dependent one-way pointers. All references are binary.

7 Object class

An object class can contain attributes (single- or multi-valued) and references and can be associated to other object classes by name bindings.

8 Interface

The communication paths between applications are called interfaces. Interfaces can exist between applications within one system or of different systems. Interfaces are binary. There can be several interfaces for each reference point, and vice versa. Interfaces can be one-way or two-way.

9 Activity

An activity is a task which is carried out within one organisation unit (at the lowest formal level of the organisation hierarchy) without awaiting communication from other organisation units. The activities of the systems planning are really activity types.

10 System activation

A system activation is the use of a system by an activity (or other system activation) which provides results to the same or other activities (or system activations). The system activations in the systems planning are really system activation types.

11 Data flow

A communication path between activities and/or system activations is called a data flow. Data flows can be one-way or two-way. Data flows can convey control and is then called control flow. Data flows in the systems planning are really data flow types. While reference points can be n-ary, data flows are binary only.

12 Routine

A routine is a set of data flows between activities and system activations which follows one input data flow to the business unit. Routines are in the systems planning considered not to contain loops even if they may be visiting the same organisation unit several times. Note that routine graphs depict data flows between activities and system activations, while data flow graphs depict data flow between organisation units and systems. Routines in the systems planning are really routine types.

difficult, because enough information is wanted to make informed decisions, while we want to leave the detailed design to the following implementation projects. The cook book identifies elements which should be surveyed and analysed in order to make rational design choices.

Several terms are assigned a special meaning in our method. See Box 1 – Terminology.

2 Surveying

The objective of the surveying phase is to identify problems and opportunities with existing systems and activities.

Figure 2.1 shows the steps in the surveying phase.

2.1 Delimitation

The scope of the study is decided together with the IT co-ordinator of the business unit to be studied. This step should provide

- identification of a list of interrelated systems to be studied
- identification of the owner organisation unit of each individual system.

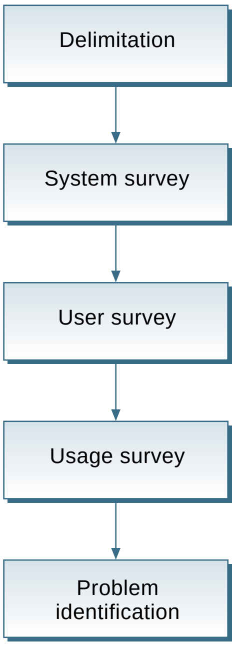


Figure 2.1 Surveying

2.2 System survey

This survey should be carried out for each system identified in the previous step. The survey can be accomplished by questionnaires to the owner organisation units. The questions comprise

- problems with existing systems and their use
- opportunities within the total application domain
- data flow between systems
- problems related to data flow
- impact of technology, organisation or market change
- user organisation units (at the lowest formal level) of the system
- identification of persons to be interviewed in each user organisation unit.

2.3 User survey

The user survey should be undertaken in each user organisation unit identified in the previous step. The survey is accomplished by interviews with the experts identified.

The user survey comprises:

- areas of responsibility of the user organisation unit
- all systems used within the entire user organisation unit
- problems related to use of each individual system
- opportunities related to these systems
- candidate manual activities to be automated
- data flow between this user organisation unit and other units
- data flow between systems; if this information is easily obtained
- problems related to data flow
- critical success factors in each organisation unit
- impact of technology, organisation and market change
- identification of real users of data managed by the systems identified in the delimitation step.

This step surveys the entire user organisation unit, and not only use of the systems identified in the delimitation phase.

The purpose of this enlarged scope of the survey is to provide information on the users' total need for system support.

The purpose of making a distinction between (system) users and data users is that the system/terminal users are frequently not the real users of data. The data users can be customers, sales personnel, technicians and managers who do not use the systems themselves.

2.4 Usage survey

This survey should be undertaken for each (kind of) data user. The survey can be accomplished by interviews of the data users. This survey need only be undertaken if the system users cannot provide satisfactory information of need for data.

The usage survey comprises:

- tasks of the data user
- data needed for the tasks
- problems related to data use
- systems providing data to the different tasks
- problems related to data delivery
- critical success factors for the data user
- impact of technology, organisation or market change.

2.5 Problem identification

This step summarises

- problems
- opportunities

within the studied application domain.

This step can be organised into three substeps:

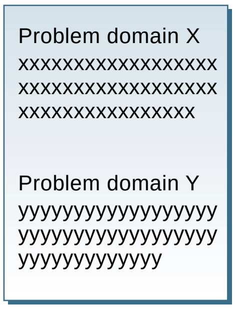


Figure 2.2 Report on project candidates

- extraction of problems and opportunities from the questionnaire replies and interview reports
- identification of project candidates from the extracted information
- organisation of the project candidates into appropriate problem domains.

The results from this step should be compiled into a report which is presented to the customer (steering group and other sponsors of the project). The results should also be validated through a hearing among system owners and interview persons. The customer makes decisions concerning immediate actions based on the results.

3 Analysis

The objective of the analysis phase is to identify a set of subjects which can be implemented as autonomous databases.

The subject analysis starts with a correction of the existing data structures. Therefore, the result of the analysis depends on the design choices made, but provides an abstraction over the data designs, as well. The subject analysis is independent of the usage of data and the external design of systems. Therefore, the subject analysis is labelled data oriented, not function oriented.

We recommend that the analysis is carried out on a per system basis, due to the size and complexity of each system.

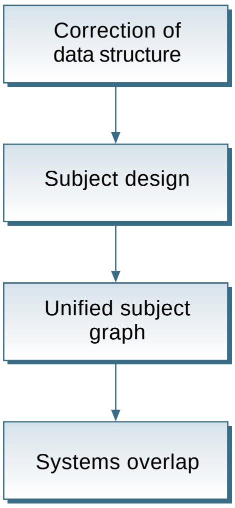


Figure 3.1 Analysis

This, however, leads to a need for harmonising subject graphs into a unified subject graph at a later stage.

The subject analysis is split into four steps, as shown in Figure 3.1.

3.1 Correction of data structure

The purpose of this step is to develop a harmonised and simplified data structure per system.

The step is carried out like a reengineering per system, and the work is based on the following information sources:

- database schema, as the primary source, if available
- system documentation
- user documentation
- user interface.

The result is documented in Graphic GDMO, ITU-T draft Rec. Z.360 [5], using

- object classes
- references
- name bindings.

Z.360 is used due to its support of name bindings and references to express dependencies between object classes. Value classes, attribute classes, structure records, etc. are removed from the graphs. Inheritance is removed by copying into the subclass.

3.2 Subject design

The purpose of this step is to design a subject graph for each data structure graph (for every system) from the previous step.

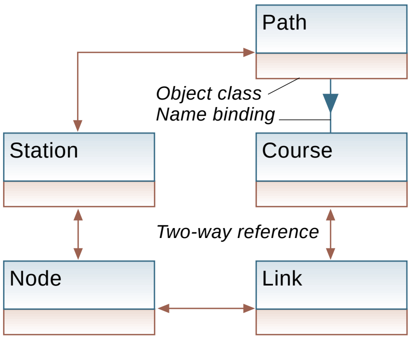


Figure 3.2 Graphic GDMO

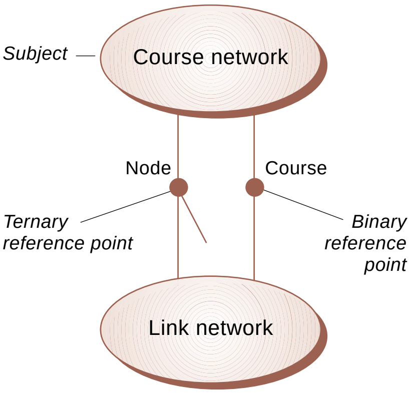


Figure 3.3 Subject graph

A subject graph comprises a set of subjects and reference points between these subjects. The subjects are disjunct aggregates of references between object classes. The references are aggregated in such a way that constraints can be enforced within a subject, and there is no need for co-ordination across subjects.

A data structure graph for a subject comprises

- references which are existentially dependent on other references within the subject
- associated references, i.e. references related to an object class within the subject and which otherwise would have been isolated in a separate subject for this reference
- object classes which are related by references within the subject
- name bindings to the mentioned object classes together with all recursively superior object classes and name bindings.

The same object classes and name bindings can belong to several subjects.

If each subject is implemented in a separate database, then global distinguished names are needed to refer between the databases. Therefore, object classes (containing these names) are used as reference points between subjects. If one object class is used to refer between more than two subjects, then this object class represents an n-ary reference point.

Note that in subject design, references are aggregated into subjects, which each in principle could form an autonomous

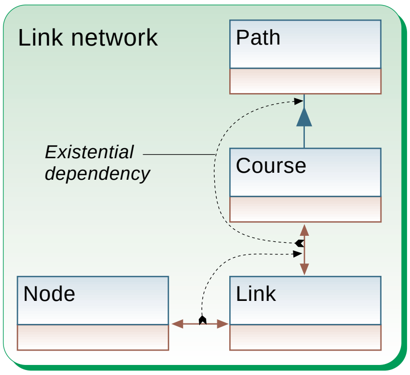


Figure 3.4 Data structure graph per subject with indication of existential dependencies

database. Subject design is, therefore, distinguishable from clustering techniques of object classes in repositories.

3.3 Unified subject graph

The objective of this step is to collect and harmonise subject graphs from all systems into one unified subject graph for the entire systems family. The systems planners must consider that

- different names may have been assigned to similar subjects in different systems
- identical names may have been assigned to different subjects in different systems
- the data structures of different systems may not be harmonised.

System	S1	S2	S3	S4	S5
Subject					
T1	x	x			x
T2		x			x
T3	x				
T4		x			
T5			x	x	x
T6				x	
T7		x		x	
T8	x	x	x		
T9			x	x	
T10	x	x	x	x	
T11					x
T12					x
T13					x

Figure 3.5 Subject-system-matrix

The unification of subject graphs will result in

- a splitting and/or grouping of subjects
- renaming and/or redefinition of subjects
- creation, deletion and/or redefinition of reference points.

3.4 Systems overlap

The objective of this step is to identify overlaps between systems.

The step can be divided into three substeps:

- make a list of all subjects of the unified subject graph
- create a subject-system-matrix
- identify and summarise overlaps.

The results from the subject analysis are collected in a report to the customer (steering group and other sponsors). The customer makes decisions based on the report.

4 Corrective design

The aim of corrective design, Figure 4.1, is to harvest immediate results from the surveying and subject analysis without having to wait for a final idealistic design.

Corrective design takes the subject-system-matrix as its starting point. Corrective design requires acquisition of knowledge from development and maintenance personnel; typically an expert per system is needed. If these experts cannot take active part in the corrective design, they have to be interviewed by systems planners.

4.1 Subject co-ordination

Subject co-ordination addresses the handling of subjects across system boundaries. This step aims at removing overlaps between systems and to improve the communication between systems.

The subject co-ordination can be divided into four substeps:

- co-ordination and consistency control
- discussion of vertical partitioning
- discussion of horizontal partitioning
- consider joining or partitioning of subjects.

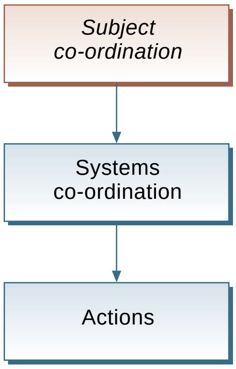


Figure 4.1 Corrective design

Co-ordination and consistency control shall

- ensure that subject and data definitions are co-ordinated and correctly interpreted across systems
- summarise and co-ordinate results across systems.

The project workers will typically encounter the following obstacles during the analysis:

- different systems use different data and subject definitions
- different systems use different or identical labels of the same or different data and subjects.

The project may find it convenient to identify a co-ordinator for each subject or a set of subjects across all systems.

Discussion of vertical overlap is undertaken for each subject which is spread across two or more systems. The sub-step comprises:

- identification of characteristics of the overlap
- assign ownership to the subject to one system
- consider if the redundancy between systems should be removed
- consider if data should be moved from one system to another
- reconsider the interfaces between systems.

Discussion of horizontal overlap is undertaken for each subject which spans two or more systems. The sub-step comprises consideration of

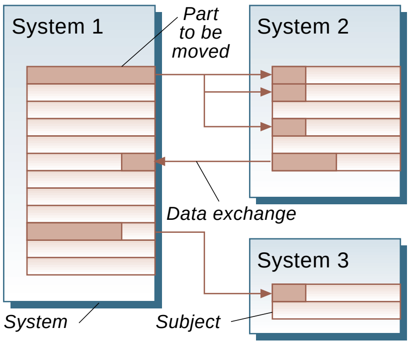


Figure 4.2 Systems co-ordination graph

- whether the span of the subject across several systems is due to horizontal partitioning of a data type
- co-ordination and communication between systems.

The last substep, ‘consider joining or partitioning of subjects’, reconsiders and summarises joining or partitioning of each subject across all systems in which it is contained.

4.2 Systems co-ordination

This step is carried out for each system and can be divided into two sub-steps:

- consideration of partitioning each system into applications
- summarising the recommendations for each system.

The project must consider if it is appropriate to carry out the first sub-step. A system can be horizontally partitioned into

- several independent and non-overlapping applications
- several autonomous, but communicating applications
- several dependent and partly overlapping applications.

The second sub-step summarises

- proposals for redistribution of functionality between systems
- proposals for changes of each system
- implications for development, administration and use of the systems.

Proposals for move of functionality between systems can be documented in a systems co-ordination graph. See Figure 4.2.

4.3 Actions

The results from Corrective design are reported to the customer and other sponsors of the project. Decisions concerning implementation must be made by the organisations responsible for each system.

5 Idealistic design

The purposes of idealistic systems planning are

- to develop an idealistic systems plan, which will serve as a vision for subsequent development projects
- to develop an application plan for each system
- to redesign activities and routines
- to develop cost and impact analyses.

The objective of the planning is to design the most efficient manual and automatic processing of data. This is achieved by moving activities as close to the customer as possible. Aggregation of subjects into systems is done in such a way that the routines are not hampered by a need to access several systems. New requirements are met by incorporating new data definitions and redesign of activities and routines.

5.1 Redesign of data and subjects

This step comprises:

- definition of data for new needs
- redesign and harmonisation of data of existing systems.

New data definitions are considered to belong to one or more hypothetical systems. This step is carried out like an iteration of the three first steps of the analysis phase. The result of the work is documented in a unified subject graph for the (new and old) data definitions.

5.2 Identification of routines

A routine comprises all data flow which results from one data flow input to a business unit until data leave this business unit.

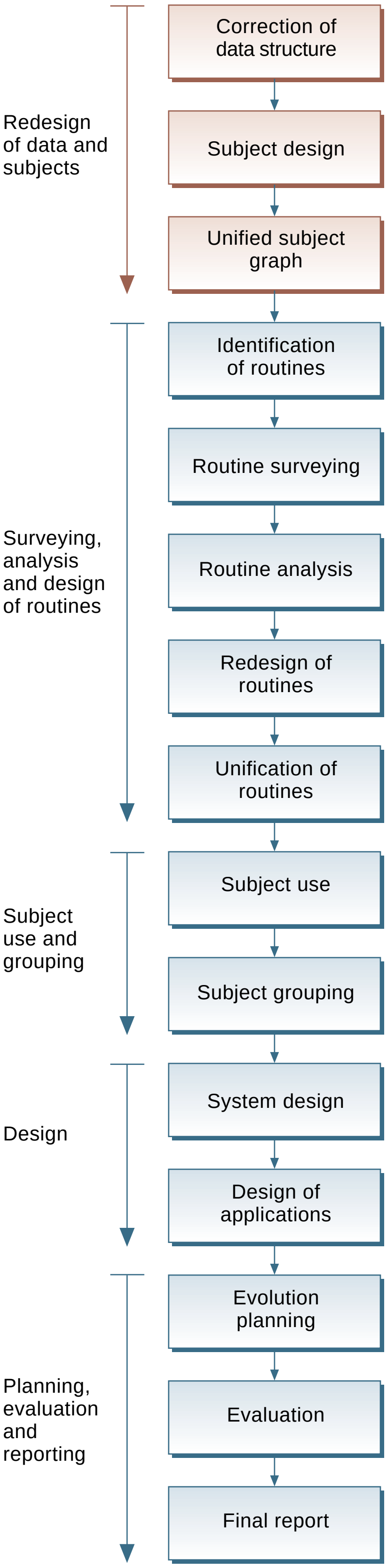


Figure 5.1 Idealistic design

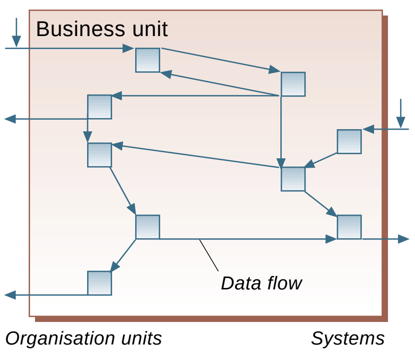


Figure 5.2 Data flow graph with initiation of two routines

A data flow graph, from the information collected during the Surveying phase, is developed – as an intermediate result – for

- all systems
- all user organisation units
- all data flow between these systems and units.

The final results from this step are

- identification of all inputs to the business unit
- prioritisation of which routines to be analysed in which sequence.

5.3 Routine surveying

Each routine is surveyed by interviewing expert users and by acquisition of routine descriptions. Note that there are frequently discrepancies between routine descriptions and the actual routine.

The routines are documented in enumerated notes. Note that a routine can visit an organisation unit several times.

The level of detail of the routine surveying has to be considered in each individual case; however, the surveying has to distinguish organisation units at the lowest formal level.

The results from the surveying can be documented in informal routine graphs.

5.4 Routine analysis

In this step a routine graph is developed for each routine. See Figure 5.3. All work carried out in one organisation unit without intermediate communication with other units is defined to be one activity.

The routine graphs depict activities, system activations (including use of manual files), and data and control flows between these. Furthermore, vertical lines between activities indicate that they are undertaken in the same organisation unit and vertical lines between system activations indicate that they are of the same system.

5.5 Redesign of routines

The purpose of this step is to design the most efficient allocation of work to organisation units and sequencing of work for each routine. This objective can typically be met by moving work as close to the initiation of the routine as possible. Note that reallocation of work frequently will imply change of access to systems.

The redesign is documented in routine graphs of the same kind as used in the routine analysis. Note that vertical lines indicate alternative work flow in case the intermediate communication (possibly via systems) to other organisation units is not needed.

5.6 Unification of routines

This step studies

- harmonisation of routines
- linking of routines

with the corresponding updating of routine graphs. Centralisation and decentralisation are essential questions of this step.

The resulting graphs are called unified routine graphs.

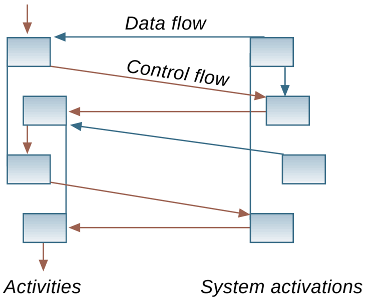


Figure 5.3 Routine graph

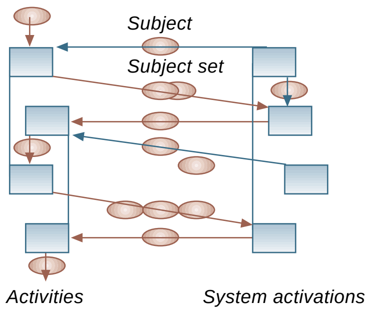


Figure 5.4 Routine-subject-graph

5.7 Subject use

In this step every subject is listed for each data flow between activities and system activation in each unified routine graph. The subjects, which are depicted in the unified routine graphs, refer to the unified subject graph.

The resulting graphs are called routine-subject-graphs. A collection of subjects can be referred to as a subject set.

5.8 Subject grouping

The purpose of this step is to identify subject groups which can serve as systems. The group of subjects used by one activity forms the starting point for the grouping. This way, work is not hampered by one activity having to access several systems.

Sometimes the tasks contained in an activity are so loosely coupled or subjects are so seldom accessed that the subjects can be organised into several subject

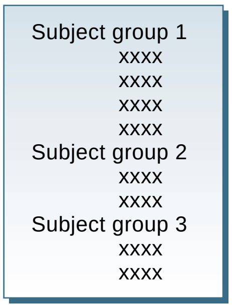


Figure 5.5 Subject groups

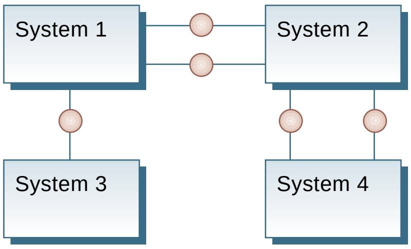


Figure 5.6 Systems graph with reference points

groups or systems per activity. We want to design disjunct subject groups or systems. Since the needs of some activities can be in conflict, some subjects may be split on several subject groups/systems or joined into several subject groups/systems. The resulting subject groups are documented in named lists and are decoupled from the activities.

5.9 Systems design

This step produces the main result from the entire systems planning work.

Based on the results from the previous step, this step proposes

- new, redesigned and existing systems
- reference points between systems.

The results are documented by

- one systems graph, see Figure 5.6
- a subject graph for each system.

5.10 Design of applications

This step will propose

- vertical and horizontal partitioning of systems into applications
- introduction of controlled redundancy between applications of each system
- introduction of controlled redundancy between systems
- interfaces and co-ordination mechanisms between applications
- interfaces and co-ordination mechanisms between systems.

The results are documented in

- an application graph for each system
- a subject graph for each application.

The project will have to consider if the tasks of this step can be postponed to the subsequent development projects.

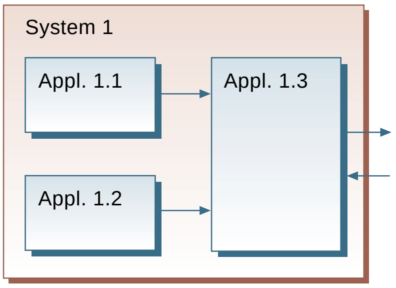


Figure 5.7 Application graph with data flow

5.11 Evolution planning

This step will propose a migration path from the current systems to implementation of the proposed systems plan.

The project will have to choose between a revolutionary or evolutionary migration strategy.

5.12 Evaluation

The objective of this step is to develop a cost-benefit evaluation of the proposals from the previous steps. This step shall produce

- cost estimates
- economic and other benefits from the proposals
- implications for personnel and other implications.

Since the idealistic plan is long range and strategic, the benefits can be long range, as well. Therefore, it may be difficult to undertake an economic analysis. In this case, it may be more appropriate to provide strategic considerations on technological, organisational, regulatory or market changes.

5.13 Final report

The results from idealistic design are collected into

- recommendations to the steering group and other interested parties
- documentation to developers.

See contents of the systems plan in section 1, Systems planning.

The results should be validated by IT support, IT co-ordinator, system owners and expert users before final decisions in the steering group or higher level bodies. The decisions should be anchored by a binding IT strategy and concrete development initiatives by the customer.

References

1 Saxlund, G et al. *The Urd Systems Planning Method.* Kjeller, Telenor R&D, 1996. (Report N 52/96.)

2 Martin, J. *Strategic Information Planning Methodologies.* Englewood Cliffs, NJ, Prentice Hall, 1989. (ISBN 0-89435-358-6.)

3 Telenor IT. *BIA : Business Information Architecture.* http://info.telenor.no.

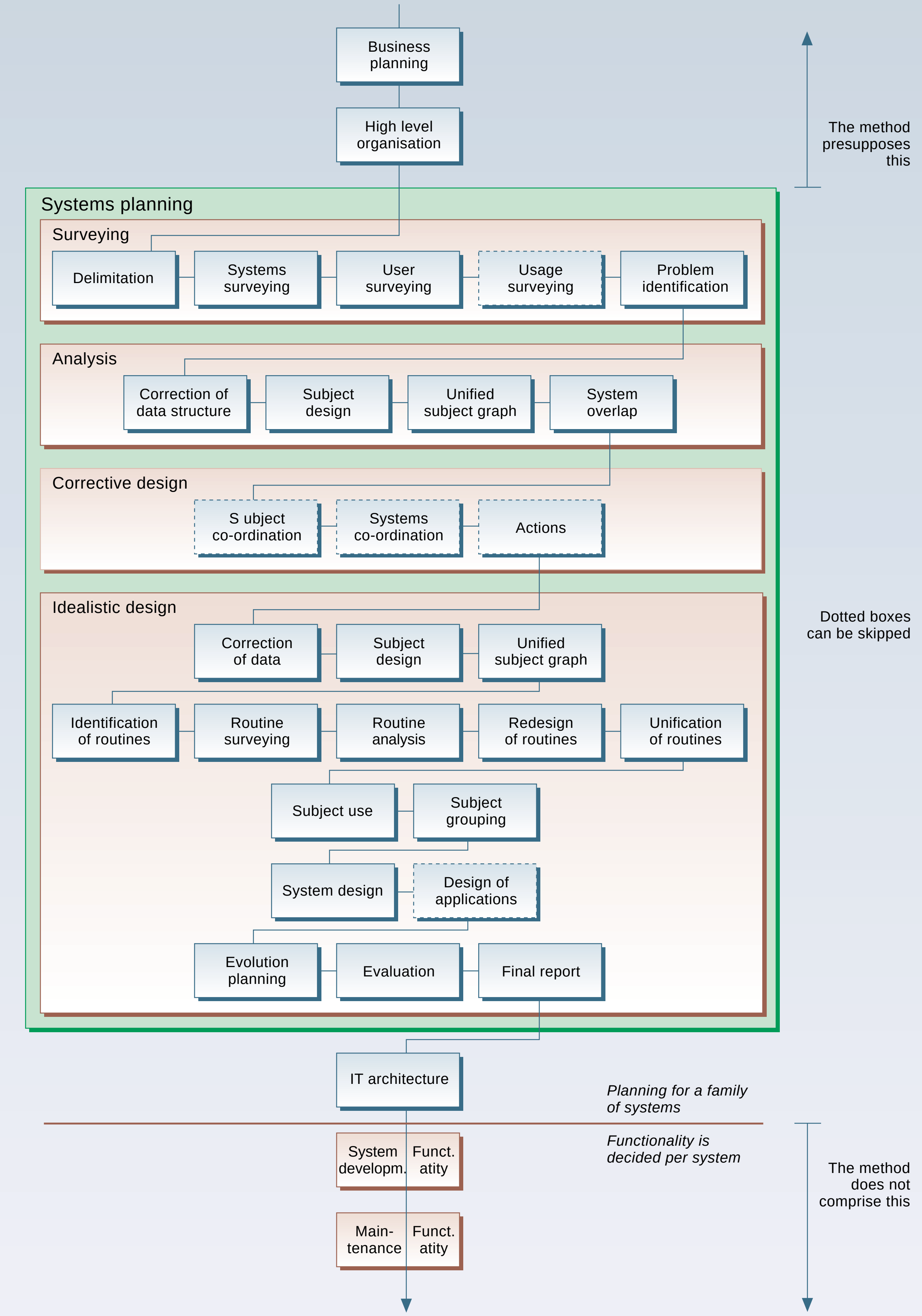
4 Furley, N. The BT Operational Support Systems Architecture Framework. *British Telecommunications Engineering,* 15 (4), 1996.

5 ITU-T. *Report of the meeting held in Geneva from 10 to 18 April 1996.* Geneva, ITU, 1996. (COM 10-R 4-E.)

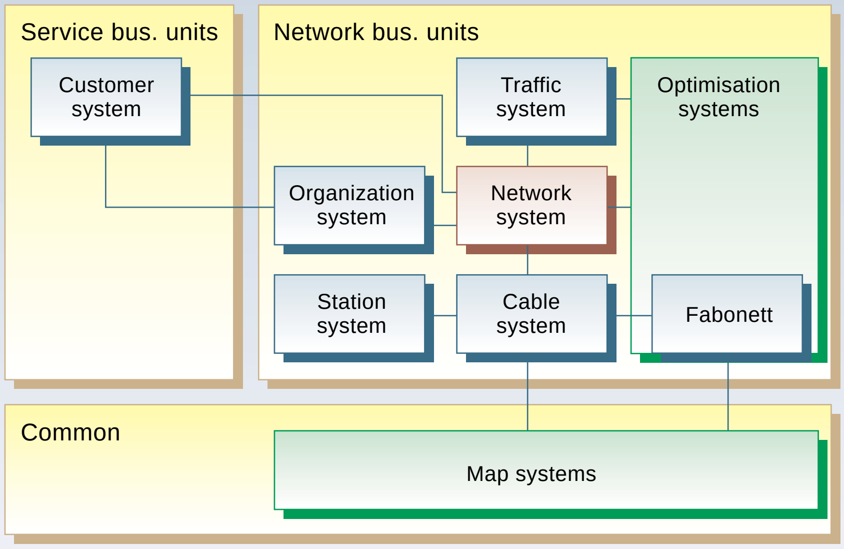
Arve Meisingset is Senior Research Scientist at Telenor R&D. He is currently working on information systems planning, formal aspects of human-computer interfaces and middleware standardisation. He is ITU-T SG10 Vice Chairman and the Telenor ITU-T technical co-ordinator.

e-mail:
arve.meisingset@fou.telenor.no

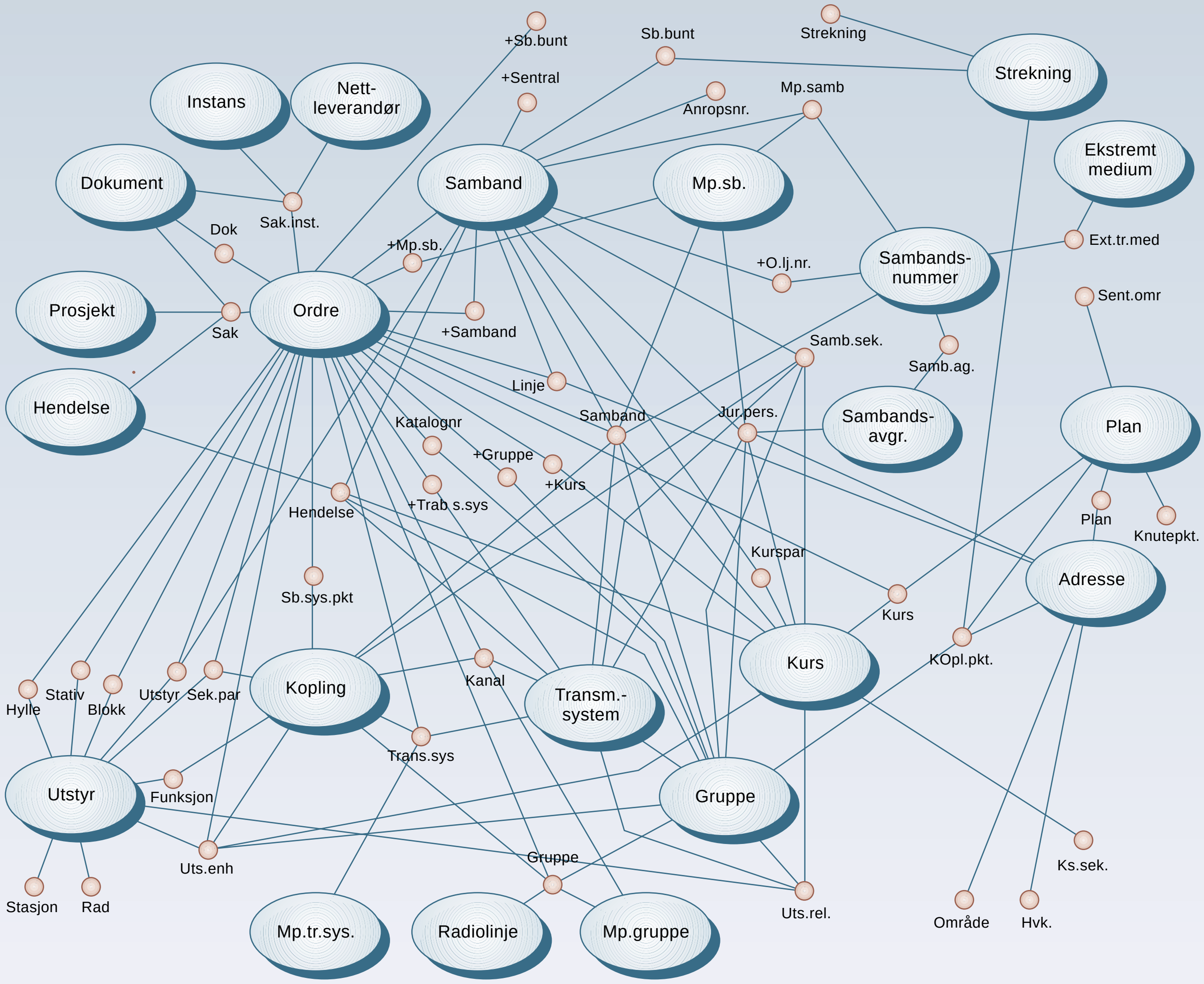
Box 2 – Method overview



Box 3 – Example Systems graph



Box 4 – Example Subject graph



Decomposition and Granularity in Systems Planning

SIGRID STEINHOLT BYGDÅS

This paper addresses the granularity problem of systems planning. Five systems planning methods are presented, and their granularity levels are identified according to a granularity level hierarchy.

In the 1970s many organisations started to computerise their data and processes. Now, twenty years later, they experience information system chaos. Purchasing of new systems seldom takes the existing systems into account, and while new computerised systems are added to the system portfolio, old systems are seldom removed. To ensure that the computerised information systems fit together and support the business in the best way possible, information systems planning should be performed.

'Strategic information systems planning', or 'systems planning' for short, has been defined as *"the process of identifying a portfolio of computer-based applications that will assist an organisation in executing its business plans and realising its business goals"* [1]. Systems planning is a complex and critical task, and it is often unsuccessful, partly because the plan is not followed [1].

The literature provides various systems planning methods, see for example [2], [3], [6] or [7]. Methods differ in scope and granularity. Some methods, like Martin [2], have the whole enterprise as its scope, while others, like the Urd method [3], have a business unit or a set of systems within the business unit as its scope.

Regardless of scope, systems planning is a complex task. To get an understanding of a complex problem area, it can be beneficial to decompose it into smaller parts that are easier to deal with. This approach is called 'top-down'. When every problem part is understood, you must put the parts together to get a new picture, which provides a better design or a better understanding of the problem area. This approach is called 'bottom-up'.

The problem of decomposition is when to stop decomposing. How do we know that we have reached the necessary level of detail? In the case of composition, the question is at which level of detail do we start? If we analyse too many details, we may get lost, and never come up with a systems plan. If we analyse too few details, we may not be able to address the real problems. This is the heart of the granularity problem. Because systems planning precedes system development, it should not require a detailed analysis of each new system. The detailed analysis should be left for the system development activity. However, for the systems planning results to be useful in subsequent system development projects, it must be so detailed that the conclusions become trustworthy. The granularity problem of systems planning can therefore be seen as how to compromise between planning efforts and quality of the resulting systems plan. There may not be one general solution to this problem, and, as this paper shows, different systems planning methods provide different answers to the granularity problem.

In this paper we present five methods for systems planning. To be able to compare the granularity levels of various methods, we have identified nine general granularity levels, see Table 1. We have arranged the granularity levels in a hierarchy, but are aware of the fact that the levels do not always define a strict scale. Another difficulty with the comparison is that even if two methods address the same granularity level, one can do this in an abstract way, while the other does it in a concrete way. Nevertheless, we find the hierarchy helpful in visualising and comparing the granularity levels of the methods. The reader should, however, be cautious when using the diagrams.

ISAC

The ISAC method was widely used in Scandinavia during the 1970s. It is in fact a system development method, which incorporates systems planning aspects as well. It is presented in this paper, because it provides a comprehensive general framework for analysing information systems.

ISAC was the name of a research group at the Royal Technical High School and the University of Stockholm, in the early

Table 1 Granularity level hierarchy

Granularity level no.	Granularity level entity class	Typical number of entities
8	Enterprise	1[1)]
7	Business Unit	5 – 100
6	Organisation Unit / System	50 – 1000[2)]
5	Activity / Function	500 – 2000[3)]
4	Subject	100 – 500[4)]
3	Entity / relationship	1000 – 2000
2	Attribute	5000 – 10,000
1	Process description	5000 – 10,000
0	Detailed implementation	5000 – 10,000

NOTES:

1) To come up with this hierarchy, we have considered an enterprise of large national and medium international size, like Telenor. Please note that the granularity levels will not always constitute a strict hierarchy.

2) We assume that each system supports the activities of one business unit only.

3) We do no generalisation of functions and activities across business units or systems.

4) A subject is a collection of relationships with corresponding peer entities. Subjects are (unlike functions) unified across systems, that means that they are only counted once even if they are used by several systems.

seventies. The group developed a method for information system development based on the work of Børje Langefors.

The scope of the ISAC method is an organisation or organisation unit. This relates to the granularity levels 6 to 8 in Table 1. Within this scope it is the people related to the organisation (unit) and the problems they experience in their work that are important. 'People' includes the following roles: user, manager, worker, customer and fonder.

The design philosophy is compositional or bottom-up in the sense that it is assumed that the solutions to subproblems within the organisation will give the solutions to the organisation's problems as a whole [5]. During the analysis, however, you will work top-down, decomposing functions and information.

The ISAC method has an initial phase (I) called Change Analysis. The purpose of this phase is to identify if the organisation needs an automated information system at all. In other words, ISAC does not, like most other methods, assume that the development of an information system is the solution to an organisation's problems [5]. Another interesting difference from most methods is, as stated by Avison and Fitzgerald in [5], that:

> *"Need is established only if it is seen that an information system benefits people in their work, so that pure financial benefit to the organisation, or some other benefit, is not thought to be an indication of need for an information system."*

The purpose of the second phase (II), activity studies, is to delimit the information system. To do this, activities that will be supported by the system are identified and analysed. Information needed by the activities, and the information flows between the activities are identified.

The third phase (III) is called information analysis. In this phase the contents and tasks of the information system are analysed. For each task, input and output information sets are analysed. (An information set is a data flow or a data store.) The analysis consists among other things of precedence and component analysis of each information set. The analysis may look quite similar to functional decomposition, but the reasoning process is different; ISAC's precedence analysis identifies the input information necessary to produce the information set studied. When the input information set is identified, the output information set derivable form this input is identified. If the two output information sets are identical, the analysis stops. To be able to reason about the transformation between input and output information sets, the structure of the information sets must be known. This structure is analysed in the component analysis. When precedence and component analysis are taken down to the lowest level of decomposition, the logic of the resulting processes (transformations) is analysed. Descriptions of the information processes are then made. A Process description is a detailed description of the prerequisites, conditions, permitted states for conditions and required actions for the process.

In the ISAC method, an information system is not necessarily computerised. In the fourth phase (IV) of the method, it is decided for each process, if it will be automated or not. Related automated or manual processes are grouped. A group of automated processes defines a program. To perform data program design, Jackson Structured Programming (JSP) is recommended. ISAC also recommends that the workers design their own manual routines. The elementary information sets that are a result of phase III, are aggregated into permanent or temporal data sets, based on their functional dependencies, and messages, files or databases are designed.

In the method's last phase (V), the equipment-independent system design from phase four is adapted to fit the particular equipment chosen.

To summarise, the ISAC method addresses systems planning aspects, and ends with a detailed system development. Figure 1 illustrates the process. Bars illustrate granularity levels, parentheses illustrate alternatives (that is and/or alternatives), and arrows illustrate transitions.

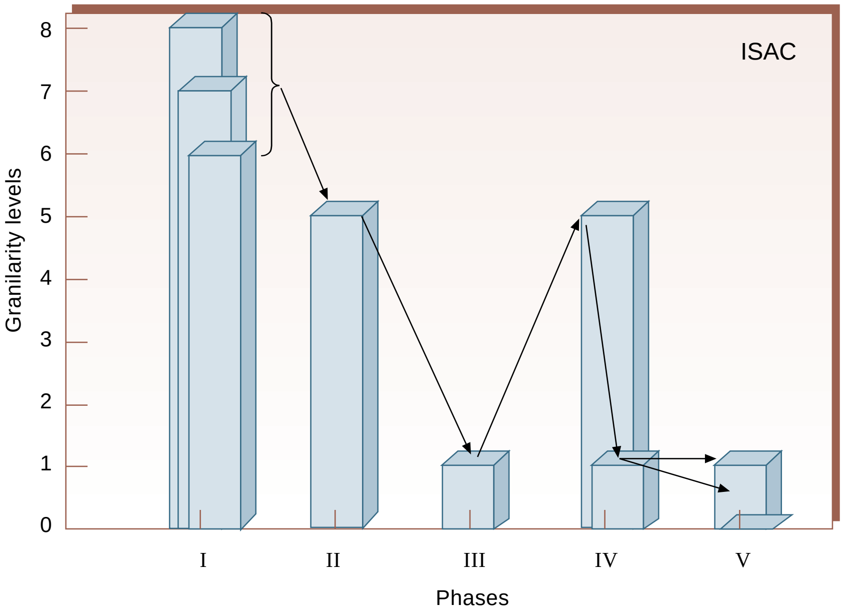


Figure 1 Granularity levels in the different phases of the ISAC method Bars illustrate granularity levels, parentheses illustrate alternatives (that is and/or alternatives), and arrows illustrate transitions.

Urd

In 1993 the Telenor Network business unit asked Telenor R&D to develop an idealistic systems plan for the key information systems of the business unit. The project started out searching for a suitable method in literature, but did not find one single method that fulfilled the needs. Some methods, like Inmon's Information Paradigm [4] was considered to be more suitable for financial than telecommunication information systems. This was due to the fact that operational data in telecommunication systems are more complex to handle than Inmon suggests, and therefore his categories provide inappropriate splits of data. As an example, several points of time must be handled to establish a circuit, and this data may not be split on several databases. Other methods, like Martin's Information Systems Planning [2], was considered to be too detailed for the purpose of this project. Inspired by literature, the project developed its own method, called the Urd method. Urd is described in more detail elsewhere in this issue of *Telektronikk*.

According to the Urd method, a systems plan should give answers to the following important questions: which systems and data do a business unit really need, and how should the data and systems be organised to support the business unit in the best possible way?

The Urd method focuses on a family of concerned systems within a business unit. Surveying and analysing are performed top-down and the design bottom-up. During the analysis existing systems are decomposed into data objects and relations, and grouped into subjects (for more details see [3]). The subjects are used to compose new systems in the idealistic design phase.

The purpose of the first phase (I), surveying, is to identify problems and opportunities related to the systems. System owners, system users and data users are interviewed during this phase. Beside the systems, information flow between systems and between systems and organisation units are studied. The lowest granularity level of this phase is systems and organisation units.

The second phase (II), data subject analysis, is performed to identify overlaps of the systems data. Entities and relationships are studied and grouped into subjects, see [3] for a definition of subject and more information on the grouping. The subjects are used as building blocks when new systems are identified, in the last phase (IV).

Corrective systems planning, the third phase (III), is optional. The purpose of this phase is to reduce overlap in existing systems, while waiting for the realisation of a more idealistic systems plan. According to the method, entities and relationships are the correct granularity level of this phase. In practice, however, the project experienced that attributes had to be studied to be able to decide if a subject overlap indicated a real overlap in data instances or not. The reason why this detailed level is necessary is because the analysis results now are used in design of improvements of current systems. A graph showing data flow between systems and ownership of subjects summarises the results of this phase.

The last phase (IV) of the method is called idealistic systems planning. A new systems plan for the application area of the system family studied is constructed. The intention is to support the business unit in the best possible way. Routines are analysed and redesigned. Subjects are assigned to redesigned activities (that are analysed at organisation level only). Systems are then designed as the set of subjects that serve the redesigned activities in the best possible way. Finally, cost-benefit analysis and migration plans are developed.

To summarise, the Urd method is only a systems planning method, even though some system design aspects are addressed in the optional third phase. The granularity levels are illustrated in Figure 2.

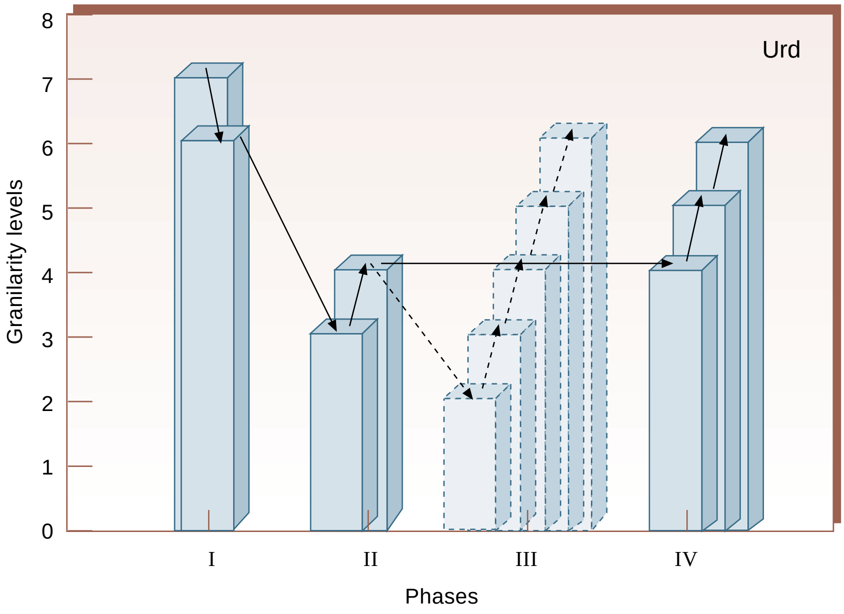


Figure 2 Granularity levels in the different phases of the Urd method. Phase 3 is optional

Information Strategy Planning

James Martin presents a method for information strategy planning in his book *Strategic Information Planning Methodologies* [2]. The scope of this method is the whole business or enterprise. The method is top-down. Involvement of top-management in the planning process is regarded as crucial for the ability to implement the systems plan. Top management is directly involved in performing the first four phases of the method. The method consists of six phases, presented below. The business areas, which are identified in the last phase of the method, are subjects for further planning studies.

The phases are:

I *Linkage Analysis Planning*. The purpose of this phase is to formulate a strategic business vision.

II *Entity-Relationship Modelling*. This phase is performed to get a better understanding of the enterprise. In addition it provides a complete list of the functional areas, functions and activities that constitute the enterprise.

III *Technology Impact Analysis.* The purpose here is to give top management an overview of potential opportunities and threats due to technological changes[1].

IV *Critical Success Factor Analysis.* Through this phase focus is put on the most important activities for the enterprise. Based on the results from this phase, decision support systems for the enterprise should be designed.

V *Goal and Problem Analysis.* In this phase the information needs of the organisation units within the enterprise are identified.

VI *Business Area Identification.* Business Areas of the enterprise, and the data and functions that belong to each business area are identified. Overlap in data and functions between business areas are minimised.

The Information Strategy Planning method does not deliver a complete systems plan. Systems plans are prepared by performing an analysis of each of the business areas identified during the last step of this method. We may therefore look at the method as a prescription of an initial part of the systems planning process.

Information Co-ordination

In his book, *Information Co-ordination – the management of information models, systems and organisations* [6], Richard Veryard presents an organic planning method. The term *organic* refers to the idea of gradual change towards a more idealistic solution, see [6]. Veryard prefers to call the result of this method a strategy, and not a plan. The scope of the method is the business/enterprise. The phases of this organic planning method are:

I Discover business objectives and goals. This is performed through a discussion with top management, which also aims to identify the business' strengths, weaknesses, threats and possibilities.

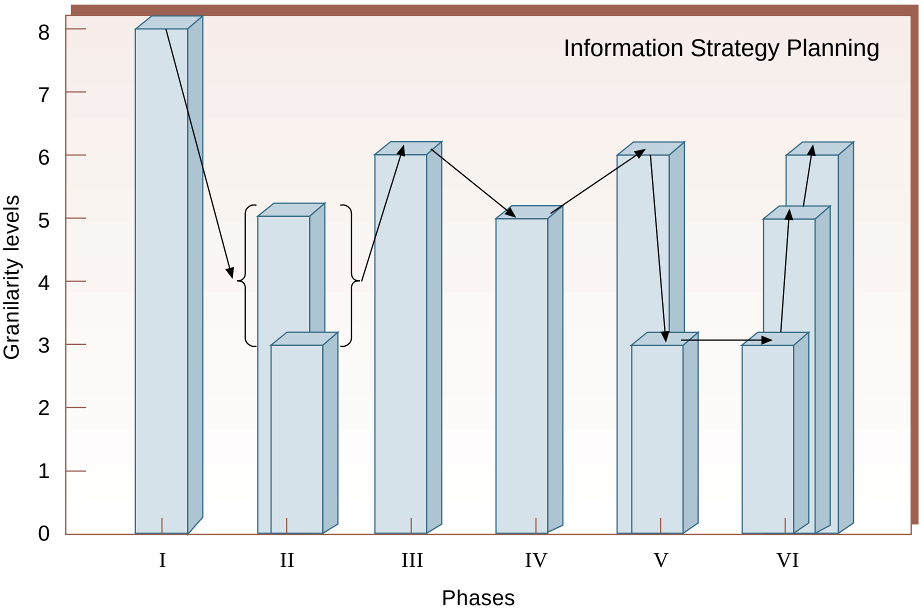


Figure 3 Granularity levels in the different phases of the Information Strategy Planning method

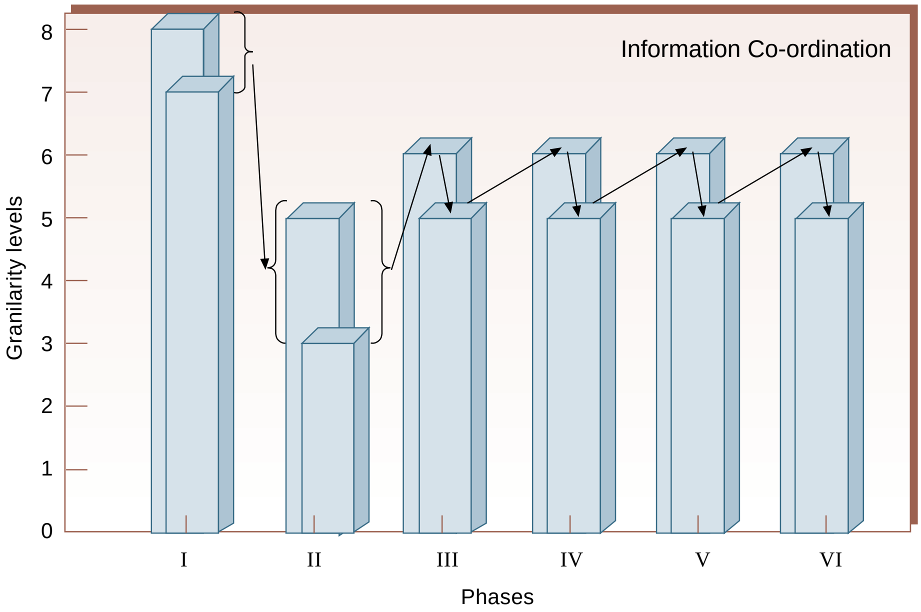


Figure 4 Granularity levels in the different phases of the Information Co-ordination method

[1] *Our granularity level hierarchy does not cover technology, like new hardware components. We have given this phase granularity level 6, only because the discussions are performed at the organisation unit level.*

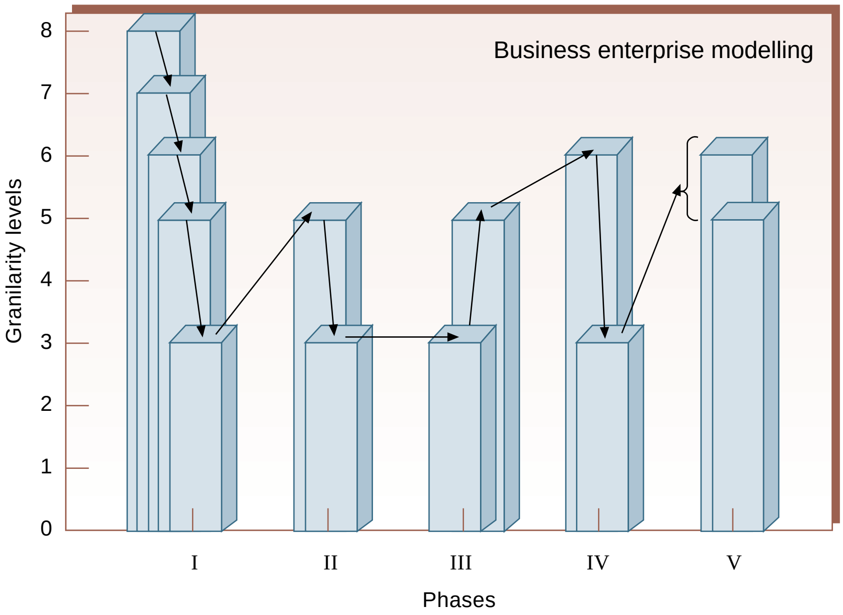


Figure 5 Granularity levels in the different phases of the Business/enterprise modelling method

II Build a high-level conceptual model of the business' functions and entities. This model is called *information architecture*.

III Survey existing and planned systems, and the interfaces between them. This includes comparison with the information architecture to ensure quality and usefulness, and discussions with top management to identify the main problems and opportunities.

IV Establish a set of system development principles and procedures, and establish the support groups and technical infrastructure necessary to support them.

V Identify and prioritise between some important projects, necessary to perform the next three to five years.

VI Tactical planning, that is identify some small and medium sized projects to be performed during the first year (in parallel with the more important ones identified above).

This method addresses systems planning only, and not system development. The systems plan is of a tactical, not idealistic, nature. We define tactical planning to have a perspective of three to five years, while idealistic planning has a perspective of five or more years.

Business/enterprise modelling

Katz presents a way of doing business or enterprise modelling [7] where one goal is to minimise data sharing between work processes. Another goal is to ensure that data users are satisfied with data quality. The scope of the method is the business or enterprise. The first part of the work, identifying all processes of the enterprise, can be performed top-down by decomposing the business functions. The last part of the work is performed bottom-up. This includes clustering data to minimise data sharing between processes. In his paper [7], Katz does not divide his method into distinct phases. To be able to compare the granularity of his method with the others, we have identified five phases, as follows:

I *Process identification.* This includes decomposition of business functions and identification of the data that each business process uses.

II *Interviews.* This phase involves employees from all over the enterprise. One of the goals of the interviews is to find out if data users are satisfied with data quality or not.

III *Design of an integrated systems architecture.* This phase includes clustering data to identify business processes. The data is grouped in a way that minimises data sharing between business processes.

IV *Simulation.* This is performed to reveal which of the business processes that will benefit the most from the suggested changes.

V *Tactical project identification.* The projects necessary to implement the integrated information systems architecture are identified.

The business/enterprise modelling method is a systems planning method only, and not a system development plan. We have presented the method as it will be performed if an integrated systems architecture is the chosen solution. This will not always be the case, see [7].

Conclusions

The granularity level diagrams show that the method with the lowest granularity level (level 0) is the ISAC method. The reason for this is that ISAC includes system development, and this requires more details than systems planning. We find the next granularity level up (level 2) in the Urd method. This is because the third phase of Urd (the optional one) contains system design aspects, which belong to system development as well. For the other methods, granularity level 3, entity/relationship, seems to provide the sufficient details. This is true for the Urd method, too, when the optional phase is skipped.

In Urd, level 3 is used to design subjects at level 4. The idealistic design is then based on level 4. The other methods presented in this paper do not address level 4. Constructing a subject can be seen as a parallel to normalising relations due to functional dependencies in the relational model. The difference is that while normalisation in the relational model focuses on functional dependencies between attributes, Urd focuses on functional dependencies between relationships. The third phase of the business/ enterprise modelling method includes clustering, and is probably based on similar ideas about functional dependencies as the relational model.

The diagrams show that all the methods, except Urd, start with granularity level 8, the enterprise level. The reason for this is that the scope of this method is a systems family of one of the enterprise's organisation units, and not the enterprise as a whole, as for the other methods. This method is the only idealistic or strategic systems planning method of the five methods presented. It may also be an operational systems planning method (with a one-year perspective), if the third phase is performed. The information coordination and the business/enterprise modelling methods are tactical systems planning methods. The information strategy planning method does not deliver a systems plan. This method constitutes a framework for organising data and functions in an enterprise.

Systems planning methods differ, in scope and granularity. Four of the methods presented in this paper go through exactly the same granularity levels, but in different order and through a different number of phases. It is nevertheless significant which of the methods we use. As we have just stated, the methods serve different purposes. When choosing a systems planning method, the most important thing to do is to ensure that the method we choose is able to answer the problem we want to solve. If we need a framework, like the one the information strategy planning provides, we do not choose Urd. Do we need a systems plan, we do not choose to do information strategy planning. And, if we need a systems plan, we have to find out if it should be an operational, tactical or idealistic plan.

References

1 Lederer, A L, Salmela, H. Toward a theory of strategic information systems planning. *Journal of Strategic Information systems,* 5, 237–253, 1996.

2 Martin, J, Leben, J. *Strategic Information Planning Methodologies.* Englewood Cliffs, NJ, Prentice Hall, 1989. ISBN 0-89435-358-6.

3 Saxlund, G et al. *The Urd systems planning method.* Kjeller, Telenor FoU, 1996. (Report FoU N 52/96.)

4 Inmon, W H. *Data architecture : the information paradigm*, 2nd ed. Boston, Mass., QED Technical Publishing Group, 1992. ISBN 0-13-850538-1.

5 Avison, D E, Fitzgerald, G. *Information Systems Development : Methodologies, Techniques and Tools.* Oxford, Blackwell Scientific Publications, 1988. ISBN 0-632-01645-0.

6 Veryard, R. *Information coordination : the management of information models, systems and organizations.* Englewood Cliffs, NJ, Prentice Hall, 1994. ISBN 0-130099243-7.

7 Katz, R L. Business/enterprise modelling. *IBM Systems Journal,* 29, 509–525, 1990.

Sigrid Steinholt Bygdås has been employed by Telenor R&D as Research Scientist since 1992. She has been working with models and methods for systems planning, system development and CASE tool evaluation. Her current interests are in systems plan implementation and software architecture.

e-mail:
sigrid.bygdas@fou.telenor.no

The Role of Subject Graphs

ARNE SOLEVÅG HATLEN

This paper presents the notion of subject graphs, which is a way of partitioning a data structure into candidate autonomous databases. Subject graphs are used in the Urd method – a systems planning method. An overview of Urd can be found in another paper in this issue of *Telektronikk*.

Purpose and overall process

In the Urd method, the term 'system' is defined to be the data that is enforced as one consistent whole. This means that a system contains all the data needed for an application area, and the rules of constraints needed to maintain consistency between these data. This definition of *system* insists on consistency within the system's boundaries, but a system cannot guarantee complete consistency of data that is communicated between different systems.

According to Urd, the first step in the process of making a systems plan is to survey the applications in the application area. The survey consists of interviewing the application owners, users of the applications and users of the data extracted from each application. The goal is to identify problems and possibilities with the old applications and usage of the applications.

The subsequent step is to analyse the identified applications. The goal is to identify overlap between systems, i.e. common data across two or more systems. To achieve this, *subject graphs* are constructed.

To design subject graphs, we start with a data structure graph for each system. Some of the systems do not have a documented graph, and for these a graph must be developed. Other systems do have a data structure graph, but not in a suitable form. Often, the data definitions of a system are found in various forms. Examples could be anything from database scripts, conversion documentation, user documentation, to plans for a new system. The documentation may not be consistent and is frequently not up to date. To analyse the data structure, all the documentation must be transformed into the same notation.

An alphanumeric notation is not very suitable for getting an overview of the data definitions, thus a graphical notation is required. The notation used is a subset of Graphic GDMO (Guidelines for the Definition of Managed Objects) [4]. Figure 1 shows a small data structure graph expressed in Graphic GDMO, showing examples of all the constructs that we use.

Note especially that there are no attributes in the data structure graph. Very often multivalued attribute groups are separated as different entities. These attribute groups are removed from the graph. This way, we get a more compact data structure graph, focusing on the central object classes.

Once a system's data structure is provided in a common notation, it is possible to perform a subject analysis on the data structure. The result is a subject

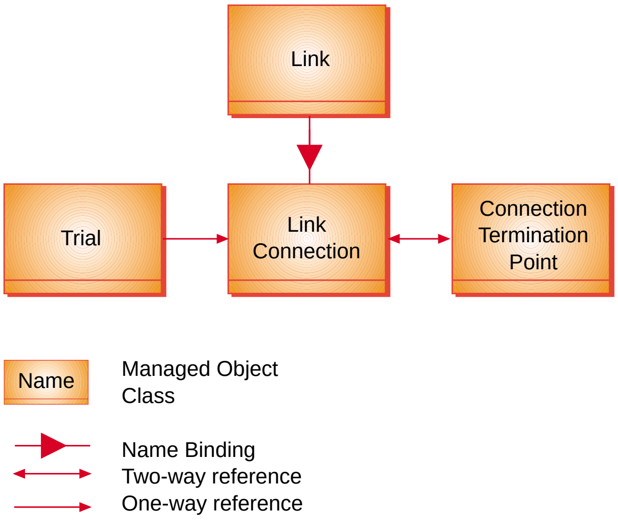


Figure 1 A small example data structure graph, showing the subset of Graphic GDMO that we use. At the bottom of the figure is a legend explaining the symbols used in the graph

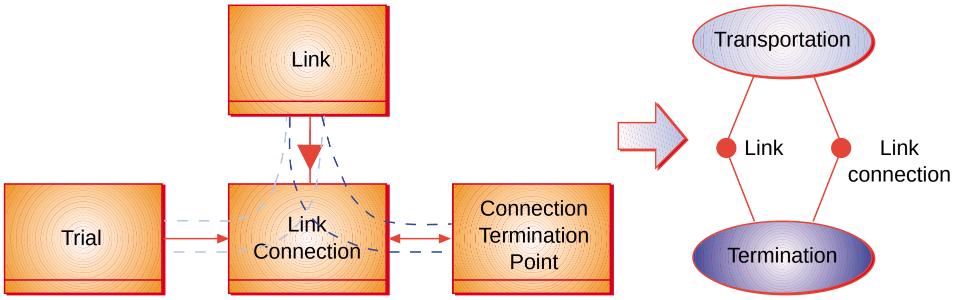
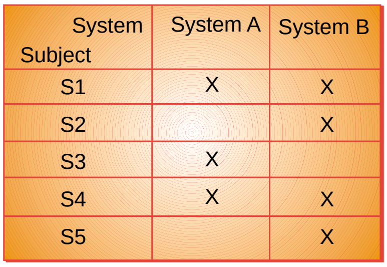


Figure 2 To the left the data structure graph presented in Figure 1 is shown, with indications of subject definitions (the two coloured areas). The corresponding subject graph is shown to the right. The subject graph consists of two subjects, Transportation *and* Termination. *The names are 'arbitrary' names, indicating the type of data the subjects consist of. Link and Link Connection are the two object classes that are needed in both subjects because of the name binding between them. Thus, they define reference points between the two subjects*



System Subject	System A	System B
S1	X	X
S2		X
S3	X	
S4	X	X
S5		X

Figure 3 An example subject-system matrix, showing five subjects, of which one (S3) exists in system A only, two subjects (S2 and S5) exist in system B only, and two subjects (S1 and S4) indicate an overlap, as they are present in both systems

graph per system. The transition from the data structure graph in Figure 1 to a subject graph is shown in Figure 2.

Having performed a subject analysis for all relevant applications, the resulting subjects from all the subject graphs are collected in a subject-system matrix, as shown in Figure 3. If two subject graphs contain the same or very similar subjects, this is considered a subject overlap.

Subset of Graphic GDMO

A specification using GDMO or another alphanumeric notation is usually very elaborate and difficult to overview. Graphic GDMO provides the overview for a GDMO specification. Graphic GDMO does not show the complete GDMO specification, however, and thus it can only be used as a supplement to the alphanumeric specification, and is indeed meant only as a supplement.

For systems planning, however, the level of details provided by Graphic GDMO suffices.

Graphic GDMO is used because of its distinction between name bindings (containment) and references. In addition to the symbols presented in Figure 1, Graphic GDMO has constructs for viewing attributes of managed object classes, and relations between managed object classes, such as inheritance (derived from), containment (name bindings), references and relationships with constraints. The complete specification of Graphic GDMO is defined in ITU-T recommendation Z.360 [4].

Subject graphs

A subject is a collection of associations (references, relationships and name bindings) between object classes. A subject graph is a collection of subjects, each covering a connected fragment of a system's data structure, with reference points between them.

Each reference or relationship can only participate in one subject. This rule does not apply to name bindings.

Between the subjects are reference points. A reference point is an object class that forms one end of two or more associations located in different subjects. A reference point can be binary or n-ary, i.e. an object class is related to two or more subjects.

Figure 4 shows a generic example of a subject graph. Here, subjects ***S2 … S5*** contain one or more relationships in which object class ***A*** takes part. The object class ***A*** is said to be a reference point between subjects ***S2 ... S5***. In the same manner, the object class ***B*** is a reference point between ***S1*** and ***S5***, and ***C*** between ***S2*** and ***S3***.

Let ***A*** be an object class participating in more than one relationship within subject ***S3***. This multiplicity does not change the nature of the reference point – ***S3*** still only 'participates once' in reference point ***A***. Thus, reference point A defines the data that must be communicated between the two subjects, if the subjects are included in different systems. It does not, however, define the data that must be communicated to perform a given operation, or the sequence of the data communication.

The purpose of defining subject graphs is to identify 'natural' vertical partitions of the data, i.e. a fragment of the data structure that in principle could be implemented as a separate database.

When designing a subject graph, each system is analysed separately from the others. The purpose of this is to keep the scope of the analysis to a manageable size. However, it is important to keep in mind that all the subject graphs are to be compared and joined across several systems. Thus, identifying similar subjects is vital.

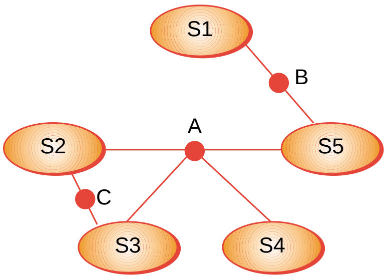


Figure 4 A generic subject graph, showing five subjects S1 ... S5, connected with three reference points A, B and C

Subject design criteria

There are few formal criteria for defining which information should be grouped into one subject.

The main criterion is to group existentially dependent associations into the same subject. This way, existential constraints can be enforced within a subject, and does not require references to other subjects. The term *existential dependency* corresponds to functional dependency in the relational model.

Another grouping criterion is that superior name bindings[1] of an object class should be present if a reference or relationship to the same object class is included in the subject. Figure 2 shows an example of this. Here, the name binding between Link and Link Connection is included in the same subject as the two-way reference between Link Connection and Connection Termination Point.

Sometimes there will be "loose" relationships (or object classes) in a data structure, i.e. a relationship that does not have any strong dependency to other parts of the data structure and does not strictly fit into any subject. These relationships could be kept as separate subjects, but may be included in a subject together with other data with which it is frequently used.

[1] *A superior name binding is also known as an identifying name binding, i.e. a name binding that is used to identify instances of an object class, without which the object instance could not be uniquely defined.*

Use of subjects

The purpose of using subject graphs is to introduce a high-level view of the data, without having to address all the details of the data structure and the specific differences of data structures within each system.

However, there are fundamental differences between subject graphs and data structure graphs. A node in the subject graph corresponds to a set of edges – associations – in the data structure graph, and an (n-ary) edge – a reference point – in the subject graph corresponds to a node (an object class) in the data structure graph. Thus, a subject graph is a dual hyper graph of the data structure graph.

The purpose of subject graphs – to define minimal candidate systems and needs for communications between these – seems to be clear. However, we have only experiences from one large project in using the graphs, and in this project we did not gain enough experience with unification across several systems. According to the experiences, we have noticed that the contents of 'similar' subjects in different subjects are most likely to be different, but having identified the subjects, it is believed to be more feasible to harmonise two subjects than harmonising two (or more) data structures.

The introduction of new technology, new services and new data standards may introduce new data structures that need to be harmonised with existing systems. Old and new data structures can be very different. Unification on subject level is then believed to be more appropriate for systems planning work, than at class level. However, for providing the unification of subjects, the discussion and detailed comparison have to be made at class level.

In the Urd method, subject graphs are developed per system, analysed for overlap and then unified across the system. This is not the only conceivable approach, as the data structures could have been unified before development of the unified subject graphs. However, this would require working with more detailed designs during the systems planning work.

Comparison with other kinds of grouping

When doing strategic planning, we seek to avoid too many details. In some techniques, such as [2], this is achieved by abstracting similar object classes into one abstract object class, often called a 'central' object class. It is stated that this abstract object class represents all objects with the same common factors. This is not the case with subjects, as subjects are collections of associations, where the content of the collection is known.

The Urd method is to be used in strategic planning. The bottom-up approach of Urd is another difference from the top-down approach of some strategic planning work. By using subject graphs, a bottom-up approach is forced, requiring the designers to identify the real data types (object classes) and associations in an application area, instead of using generalised, i.e. less detailed abstractions. It is believed that a bottom-up approach at this level yields a better result.

In [1], William Inmon suggests a method for collecting data from an organisation's computer systems. Inmon's procedure suggests grouping the data according to their usage. The data are divided into two categories, one for operational data, and another for data used in decision support. The latter category is then divided into atomic, departmental and individual data.

The goal of the grouping is to enable usage of data for new purposes. Inmon's grouping differs from the grouping of data into subjects, in at least two ways: First, the data in a subject are grouped according to usage. Second, the subjects do not necessary become databases, several subjects can later be joined into one database.

In [2], Martin and Leben describe methods for strategic information planning, and techniques to support the activities in this method. They define entity-relation (ER) models associating functions with business units and data entities. As with Inmon, these criteria for grouping entities differ from the criteria for grouping information into subjects, in that subjects are organisation independent. However, in an idealistic design phase of the Urd method, subjects are grouped into systems according to their usage in manual routines.

Clustering [7, 8] is a much-used term for techniques for grouping data. Clustering might look similar to defining subject as it defines clusters of data items used in the same operations. However, the objective is typically to minimise the number of page references in a disk operation, or to localise object instances that are used in the same transactions. In both cases, the objective is to increase the performance of a system. Clustering schemes are concerned with object instances and attributes, thus operating on a more detailed level than subject graphs. Clusters also differ from subjects in that subjects are class level collections, whereas most clustering techniques group data instances. There are schemes for class level clustering, but they usually have the same goal of minimising remote references, as with instance level clustering.

When using subjects as a basis for implementing computing systems, subjects might be joined on class level and partitioned on instance level, according to some distribution criteria. Thus, clustering is used for a different purpose than subjects and gives very different results.

Usefulness

The *"Use of subjects"* sections have identified several applications for subject graphs. Identifying overlaps across system borders, supporting the harmonisation of computer systems and identification of possible partitions of distributed applications.

However, subject graphs are probably not the universal medicine for all harmonisation problems[2]. When constructing subject graphs for a set of related systems, person-dependency might be a problem. To ensure that subject graphs are comparable – and this is usually a requirement when analysing an application domain – the designers must work closely together, having at least an indication of the common fragments of the data model in different systems.

Neither does subject graphs present a general solution for the problem of schema integration or migration. It is quite possible to make data structures completely incompatible with others, and

[2] *And neither has this been the intention of subject graphs or Urd.*

mapping fragments into subjects does not present a solution for integrating two different models, or migrating data from one data model to another model.

We have, however, good reasons to believe that construction of subject graphs can be helpful in giving a simplified view of possible overlaps across systems within an application area. With the bottom-up nature of constructing subject graphs one could be more confident of the completeness of the overview, i.e. that every part of the systems' data structures are included, than with a top-down approach.

References

1 Inmon, W H. *Data architecture : the information paradigm*, 2nd ed. Boston, Mass., QED Technical Publishing Group, 1992. ISBN 0–99435-358-6.

2 Martin, J, Leben, J. *Strategic Information Planning Methodologies.* Englewood Cliffs, NJ, Prentice Hall, 1989. ISBN-0-13-850538-1.

3 Meisingset, A. Graphic GDMO. *Telektronikk,* 93, (2), 94–96, 1997.

4 ITU-T. *A Graphic GDMO.* Geneva, ITU. (ITU-T Recommendation Z.360.)

5 ITU-T. *Information Technology : Open Systems Interconnection : Structure of Management Information : Guidelines for the Definition of Management Objects.* Geneva, ITU, 1992. (ITU-T Recommendation X.722.)

6 ITU-T. *Information Technology : Open Systems Interconnection : Structure of Management Information : General Relationship Model.* Geneva, ITU. (ITU-T Recommendation X.725.)

7 Meisingset, A. The Urd method. *Telektronikk,* 94 (1), 12–21, 1998 (this issue).

8 Everitt, B R. *Cluster analysis.* Halstad Press, 1993.

9 Javin, A K, Dubes, R C. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

Arne Solevåg Hatlen is Research Scientist at Telenor R&D, where he has been involved in systems planning and systems planning methodology work in the area of Telecommunications Management Network (TMN). His main interests are object-orientation in design and implementation, and he is currently working in a Telenor R&D project dealing with the object web and middleware technologies for the Internet.

e-mail: arne.hatlen@fou.telenor.no

Three Perspectives on Information Systems Architecture

ARVE MEISINGSET

This paper proposes three perspectives on the end user data managed and provided by a computer system. The proposed perspectives are normative and not descriptive, as current practice frequently deviate much from the proposed norm. The purpose of the proposal is to present principles which can serve as subject for discussion as well as guidelines for design and use.

Introduction

The end user data define what the system looks like as seen from the outside. The three perspectives define how these data relate to the application domain, the end user organisation, and how the data relate to each other. Hence, the perspectives provide theories for how a system can relate to the outside world and how the system of data can be structured.

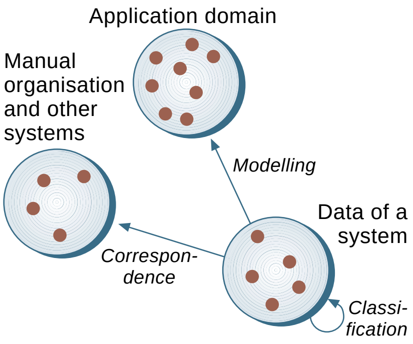


Figure 1 Perspectives on information systems

The modelling dimension states how data relate to entities in the application domain, e.g. the telecommunication network, being managed by the data of the system, see Figure 1. The correspondence dimension states how the organisation of data into systems, functions and screens relate to business units, organisation units and tasks. The classification dimension states how data relate to their data definitions, meta data definitions, etc. The reader should be warned that many approaches provide neither modelling nor correspondence, even if they (inappropriately) use terms like 'models' and 'tasks'. Also, there are many alternative (mis)conceptions about classification and instantiation.

Each of the subsequent sections provides theories for the three dimensions. Even if the sections provide a technical definition of the three dimensions, the reader should note that they define profound characteristics of a system in an application and usage context.

Figure 2 provides a framework for interpretation of the dimensions. The framework defines the formats needed for the mapping of data between two media. Not all formats and all transformations are needed in every information system (IS), however, it is these transformations which make ISes interesting and useful to most users. If data are only replicated between systems, then only the middle layer(s) of the framework is (are) needed.

A more complete exposition of the framework illustrated in Figure 2 is found in [1].

The reader should note that the correspondence theory of human-computer interface design prescribes an organisation dependence between organisations and systems. This theory is contradictory to theories of virtual organisations claiming independence of locations and organisational boundaries. The organisation dependence can be illustrated by the following realistic example: Suppose a telecommunication corporation is split into a service business unit and a network business unit. In order to support the independence of these business units and their rights to choose products from and deliver services to competing organisations, the systems should be split on these two units – the service business unit may need a service management system, the network business unit may need a network management system, and they may exchange data automatically via an

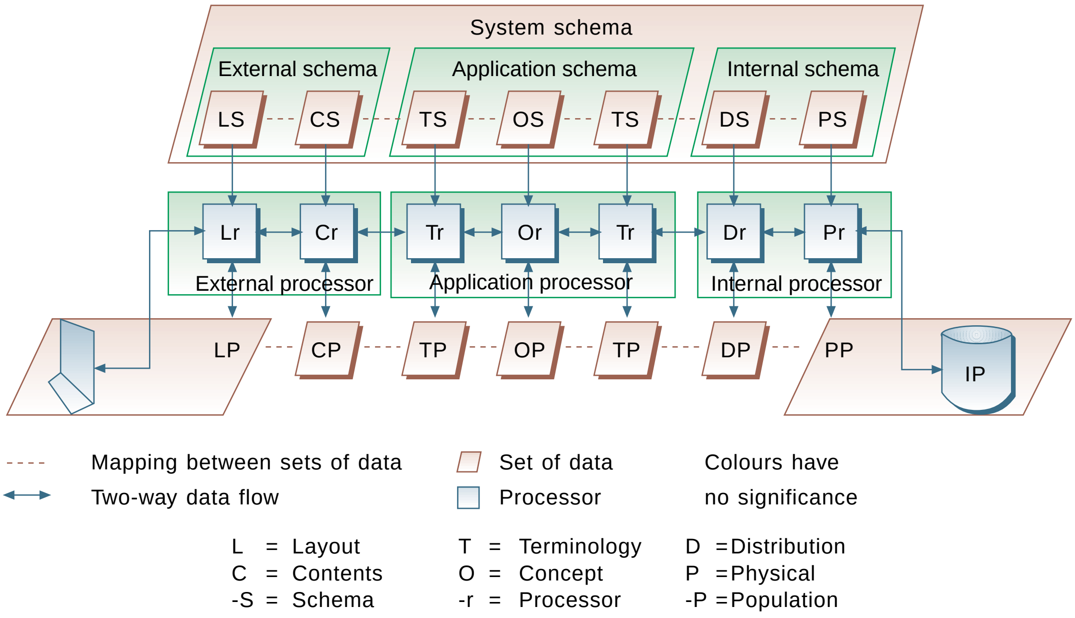


Figure 2 The data translation reference model

electronic order handling system. If, alternatively, the business units were merged, both service and network management could most efficiently be provided by one integrated system for this unified business unit, and the split between management systems was not needed. There is no generic information system architecture (ISA) or approach which is robust to such organisational changes, but there may be software architectures, tools and development techniques which can simplify reorganisation of systems when needed.

The three proposed principles provide linguistic deep structure, system theoretical contributions to human-computer interface design. The principles relate to language theory. The modelling theory relates to denotational semantics. The correspondence theory relates to correspondence semantics. The classification theory provides a copy 'semantics'. The author prefers to reserve the term 'semantics' for the denotational version only. It is outside the scope of this paper to provide parallels to philosophy, linguistics and formal language theory.

The Model Theory for Human-Computer Interaction Design

This section adapts model theory interpretation of language statements to human-computer interaction design.

Background

Statements can be categorised into the following three kinds: 1) Commands, like 'Remove the diskette!' and 'Perform x + 1'; 2) Questions, like 'What is the customer number?' and 'If x < 3'; and 3) Informative statements, like 'Customer 1 has phone number 3' and 'x = 2'.

Here we will deal only with informative statement. Informative statements state an association between terms.

Some terms or combination of terms can denote a phenomenon in some application area, e.g. the combination of terms 'circuit name A-B 1' can denote a unique circuit in a network of some administrative domain. The total expression 'circuit name A-B 1' is said to denote, model or describe this phenomenon. However, each of the component terms, A, B and 1, of the expression may not denote, model or describe any phenomenon.

Some terms or combination of terms can be created to provide an overview of other terms, e.g. 'circuit group A-B 1' may be created to provide an overview of the constituents 'circuit A-B 1', 'circuit A-B 2' and 'circuit A-B 3' between the same two nodes – A and B – in the network. The combination 'circuit group A-B 1' constitutes a derived data item and may not denote a directly observable phenomenon in the network. Therefore, this combination is said to describe/model nothing.

If data denote something, then the mapping is considered to be 1:1. In any case, there is a functional mapping (n:1) from phenomena to data. If the relationships between or behaviour of the phenomena are modelled, then the mapping is isomorphic, i.e. the modelled relationships and behaviours are preserved as associations among the data. This property of the mapping makes the data become a model of the described world.

Note that in this text we will only discuss denotation/description/modelling mappings between terms and phenomena. We will neither discuss assertion mappings from statements to propositions/facts nor the assignment of truth values derived from these mappings.

Several terminologies may be used to describe identical or overlapping worlds, e.g. 'circuit A-B 1' may denote the same phenomenon as 'samband 12345' in another terminology. The requirements for functional mappings from phenomena to any data and isomorphic mappings to describing data apply to every terminology.

Rather than introducing mappings between phenomena and data in each terminology, an intermediate conceptual level between phenomena and terms can be introduced. A functional and isomorphic mapping is introduced between phenomena and concepts at the conceptual level. Each concept can then be mapped by an isomorphic mapping to a term in some terminology. These mappings must be stated explicitly, e.g. ∃!x (Denotes (circuit A-B 1, x)∧Denotes (samband 12345, x)) states that there is exactly one x (∃!x) such that x is denoted by either of the compound terms 'circuit A-B 1' or 'samband 12345'. The two denotation mappings state the formal semantics, i.e. the common formal meaning, of the two syntactical expressions. If this mapping is not stated, then the data do not model anything in the formal sense. The data are then just inscriptions, not descriptions.

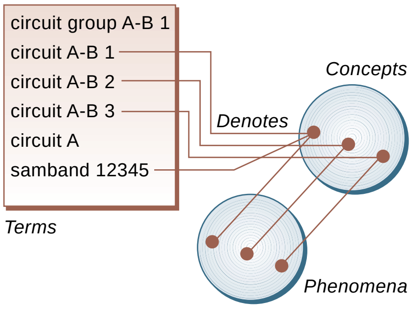


Figure 3 Use of Ogden's triangle to depict mappings between terms, concepts and phenomena. Some terms denote concepts, which model phenomena

If no formal denotational semantics is provided, the data may still be intended to model something in the informal sense. However, the data designers and end user operators must take great care to ensure that the mapping to each individual real world phenomenon is commonly understood and shared by the users of the data. This implies that

- each individual phenomenon is uniquely baptised, labelled or otherwise distinguished from other phenomena
- classes are clearly defined, explained, related to other classes and exposed to all users concerned.

Guidelines

The following modelling guidelines apply to the design of data of the application layer of the HMI reference model:

1 There should be a functional mapping from phenomena in the application area to application layer data.

2 If the application layer data describe some phenomenon, then this mapping should be isomorphic.

3 The application layer should include centralised definitions of all alphanumeric, graphical or other terminologies used within the application,

including synonymy mappings between terminologies or mappings to common concepts.

4 There should be a homomorphic (n:1) mapping from data instances to data classes in the application layer.

5 The application layer should contain all relevant derivation rules between and constraints on the data.

ITU-T Recommendation Z.352 Appendix I [2] provides additional guidelines for data design. Both basic and derived data are defined, managed and provided from some user needs. Analysis of user tasks may help to identify such needs; however, the tasks may as well be defined to manage data already defined, or the tasks may support needs outside the boundary of the task analysis. Therefore, data must be provided from a market point of view and not from a restrictive task analysis only.

The following modelling guidelines apply to the design of the contents layer of the HMI reference model:

1 The contents of screens and reports found in the contents layer should contain no other data than those defined in the application layer of the application, and permissible manipulation commands on these data. The contents of each screen and report is typically a subset of the information contained in one application system.

2 The contents of each screen and report should form one connected graph. There should be a homomorphic (n:1) mapping from the contents graph to the corresponding application layer data. See additional note in principle (5).

3 The contents of each screen and report should form a tree, possibly with references between nodes in this tree or to nodes outside the tree.

4 Any data object from the application layer can form the root of one or more contents trees.

5 Referenced data objects in the application layer may become subordinates in a corresponding contents tree. Therefore, the contents of the contents layer may not have identical structure to the corresponding subset of the application layer. However, there is no transformation of data formats between the two layers.

6 If an object is listed subordinate to its superior objects, then local distinguished names can be used. If subordinate objects are listed without their superior objects, then global distinguished names are used. However, it is permissible to list objects and their superior in reverse order, using local distinguished names only, if this is properly indicated to the user.

7 There should be a homomorphic (n:1) mapping from data instances to data classes in the contents layer.

Item (5) corresponds to the use of subclauses in natural language: The main clause refers to an item which has some characteristics, defined in the subsequent subclause. For example, ‘Circuit has End-point, which is a Station and has Name and Address’ In the application layer ‘Circuit’ and ‘Station’ are two different object classes. ‘Circuit’ has an attribute ‘End-point’ referring to ‘Station’. ‘Station’ has two attributes ‘Name’ and ‘Address’.

The given principles for contents design apply both to data classes and data instances. These principles address how informative statements of the application layer are mapped into the contents layer to maintain the information stated in the application layer.

The following modelling guidelines apply to the design of the layout at the human-computer interface:

1 The root node of the contents tree should be clearly indicated in every presentation.

2 The context in which the presentation is provided, e.g. indication of system name, function name, etc., should clearly be indicated in every presentation.

3 The layout – alphanumeric, graphic or other – should clearly indicate how all the data items relate to each other and the correct sequences of these references, e.g. <John, 12345>=/=<12345, John>.

4 The layout should clearly indicate the class of every data item presented and the relations between these classes.

5 Class labels appear as headings or icon types at the HCI.

6 References between objects should – both in alphanumeric and graphical presentations – be clearly distinguished from the presentation of the referenced object itself, such that the user can distinguish creation/modification/deletion of a reference to an object from creation/modification/deletion of the object itself.

7 Subordination, superiority and globality of objects, attributes and values should be clearly indicated.

8 Fonts, sizes and colours may be changed in the mapping from the contents layer to the layout. However, the basic data types should not be altered.

Rationale

The rationales for the given modelling principles are to make

- data understood and easy to manipulate by the users of the data
- the data definitions, structure and formats harmonised and recognisable throughout all presentations
- the structure of the data visible and recognisable to the users of the data
- the data uniquely interpretable by the users of the data
- the mappings between instances and classes visible, such that classes can be inferred from instances and vice versa

The Correspondence Theory for Human-Computer Interaction Design

This section provides a correspondence theory interpretation of language statements to human-computer interaction design.

Background

Correspondence theory is concerned with the contexts in which statements are uttered, and with the correspondence mappings between a statement and its contexts.

The statement ‘Person 1 has phone number 3’ may assert a state of affairs in some given application area. Aspects of this assertion/denotation mapping were discussed in the previous section. In this section we will discuss the contexts explaining why this statement is uttered and assign the statement to the appropriate contexts. Statements have both senders and receivers; however, this distinction will not be used in this section.

Identical statements may be stated in different contexts. The statement ‘Person 1 has phone number 3’ may be provided to a telephone user who wants to phone Person 1. The information may be provided in a paper directory, by the enquiry service, as information in an e-mail, in a technical paper, or appear in a service provision task of a telecom operator. Even if, in this case, the final use of the information is much the same, the organisational contexts in which the statement is uttered and used are very different. If identical information is given in two different administrative domains, both denotations and terminology can be different. Identical terms can refer to different instances and to different classes. Therefore, the knowledge of the correspondence to the organisational contexts is required to provide a complete and unique interpretation of the statements.

A modern organisation is considered to be an information system. This means that we will consider the organisation to consist of manual and automatic information processing. In this context we will disregard classical organisations containing labour like ditch digging, transport, production, etc. in manual or automatic form. We will constrain ourselves to manual and automatic information processing only. If other labour takes place in the organisation, we will only study the information processing part.

The correspondence theory of human-computer interaction design addresses the mapping between statement production to or from the computer and the organisation of manual tasks of the users and user organisation.

Computer systems should not be designed for a given organisation of manual work only. Also, a manual organisation should not just adapt to a given computer system. Rather, both the computer system and the surrounding manual organisation should be designed interactively, such that the best use of both automatic and manual resources could be achieved, ref. the system theoretical school for systems development.

The design of human-computer interfaces must realise the different strengths and weaknesses of automatic and manual information processing, ref. the socio-technical school for systems development.

Finally, a designer of human-computer interfaces has to realise and balance different and conflicting interests by different user groups and individuals, ref. the critical school for systems development. This last point implies that human-computer interfaces should not only be evaluated in a laboratory setting, but must be evaluated in an organisational setting of terminal operators, operator organisation, management, customers and others.

To define the user groups and stakeholders of an information system is no trivial task, and the user groups tend to change as time passes. For example, telecom service customers may at first not be considered candidate operators of the human-computer interface. However, later they may be managing their own services over the interface. Even in the first case, they are real users of the data, which the telecom terminal operators may not be.

In addition to the above concerns, the human-computer interaction designer should bear in mind which perspective on the system is imposed by alternative designs and systems development methods. A process oriented design may put users and systems in a strict work flow. Typically, this provides a high degree of automation and little flexibility. A tool perspective put the main responsibility for undertaking tasks on the users, while the system is used to support this work. The tool perspective may not provide a high degree of automation, but may provide great flexibility in work flow and use of the system. ITU-T Recommendation Z.352 [2] recommends a data-oriented approach: In this case, the computer system is an editor on data, which may model some Universe of Discourse. This is a tool perspective on human-computer interaction design. Recommendation Z.352 Annex A.3 Data design indicates how processes/functions can be automated within a pure data-oriented approach. In this case, the computer system acts as a process control system.

Principles

The following correspondence guidelines apply to the structuring of the application layer of the HMI reference model:

1 Business planning of market segments and market strategy is normally considered to be outside the scope of human-computer interaction design. However, business planning can provide premises for human-computer interaction design, and human-computer interaction opportunities may affect business planning.

2 The overall organisation of the corporation into business units is normally based on business planning, and is considered to be outside the scope of human-computer interaction design. However, business units are considered to be the legal customers of computer systems. Therefore, the scope of a computer system is defined by the individual business unit. This does not prohibit the system being used by customers outside the business unit itself. The business unit normally forms the maximum scope of interest for a systems directory of any user within the business unit.

3 A computer system is defined to enforce its data as one consistent whole, i.e. the system will guarantee that a statement p and ¬p will not be kept simultaneously in a system at any moment of time. If such conflicting statements are provided, they must be provided from different systems. Occasionally, several business units may co-operate and act as a customer of one computer system, but normally a business unit will require freedom to organise its own manual and automatic information processing. This flexibility is what makes it an autonomous busi-

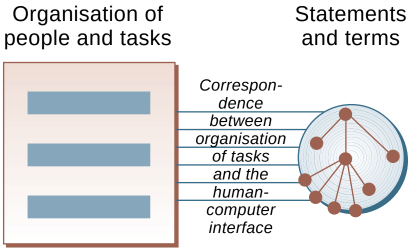

Organisation of people and tasks
Statements and terms
Correspon-dence between organisation of tasks and the human-computer interface

Figure 4 While the basic structures of statements and terms are given by their modelling roles, selection, overall structure and ordering are controlled by the organisation of people and tasks

ness unit. This independence of business units does not prohibit several business units acquiring identical systems holding different data, and systems of different business units may exchange information, but consistency of data is not guaranteed at any time. Every user should be made explicitly aware of which systems he is using and the borderlines of these systems, as this can affect his work within each system. A system can be either centralised or distributed; however, this is not relevant to the user as long as consistency throughout the entire system is enforced. If consistency is not enforced, then there are several systems.

4 Maximum flexibility of work organisation is achieved if as much work as possible is moved to the business facing office within the business unit. This redesign of work flow forms a premise for design of systems boundaries. The systems should normally be designed to support all work on a routine within one office of the business unit and minimise the need to shift between several systems to carry out this work. However, conflicting needs between several routines and offices have to be observed. This principle observes the need to take routine design as a premise for the design of systems boundaries. Intermediate organisation levels between the corporation and business units are normally not relevant to human-computer interaction design, and intermediate levels between business units and offices (i.e. the formal organisation units at the lowest level) are normally not relevant.

Note that correspondences are stated between systems and organisational units. Organisational units have real objective and observable existence, while business processes and tasks have no objective existence, unless they are defined relative to organisational units. Business processes and tasks can be considered to make up an alternative organisation structure; however, they are not authorised to constitute one. Therefore, system plans and system designs cannot be designed from the perspective of abstract business processes and tasks, but should be designed from the perspective of an existing or planned organisation of the corporation.

The following correspondence guidelines apply to the design of the contents layer of the HMI reference model:

1 One individual user or user group will typically have rights to use several functions within one system. The partition into functions should not unnecessarily hinder the user to perform several tasks simultaneously. Therefore, functions should be defined more from an access rights perspective than identification of individual tasks. However, sometimes dangerous operations may be moved into a separate function to avoid possibly harmful, but permissible operations.

2 Modelessness is achieved by not tailoring and sequencing a set of operations to a task. Modeless dialogues are created to obtain flexibility and, hence, non-correspondence between human-computer interaction designs and tasks. If greater control is wanted, then screen pictures can be ordered into one or more alternative sequences, corresponding to steps of the tasks of the individual or typical user.

3 The design of layout and contents of each screen picture is constrained by the modelling guidelines in the previous section. However, some freedom exists to further tailor the human-computer interface to the tasks. This is obtained by ordering the information in the most appropriate sequence for performing the task without violating the modelling guidelines. However, the user can also benefit from having the information presented in a harmonised sequence across several tasks. Therefore, contentious choices of trade-offs between harmonisation and tailoring must be made.

4 Great flexibility is provided to the end user if he can perform any permissible command to a data item in every place it appears. This provisioning should not be constrained by function borders.

The following correspondence guidelines apply to the design of the layout layer of the HMI reference model:

1 Great flexibility and tailoring can be achieved simultaneously by allowing the user himself to perform the tailoring. The tailoring can include means to state selection and projection of data in any screen picture and report – alphanumeric, graphic and other – and means to create minor modifications of fonts, colours, sizes and graphical layout. These changes should be local to the individual user and not affect other users or constraints or derivations enforced by the system.

Rationale

The rationales for the given correspondence principles are to provide

- correspondence to the concrete organisation of human work
- understanding of these bindings and lack of such bindings
- efficient support of the work being done
- flexibility in the organisation and undertaking of the work
- flexibility in the design of the computer system.

The Classification Theory of Human-Computer Interaction Design

This section provides a theory of how the form of data instances and classes should relate to each other from a human user point of view.

Background

Most programming and specification languages require a human or computer interpretation of the statements in those languages in order to produce or validate the permissible form of a data instance item. Typically, the prescriptions contain keywords and constraints which may not make it straightforward to validate the form of a given data item. Also, use of different word orders in the prescriptions and the corresponding data instances can contribute to these difficulties. This is exemplified in Figure 5. See details in [3] and [4].

Use of various kinds of inheritance (of Object Class, Name Binding, Relationships, Packages, Attributes and data value types) can make it very difficult to validate forms of a data instance, as pieces of the prescription are spread out on many remote statements.

The example in Figure 5 shows Object Class labels, Package labels and Attribute labels being defined in separate and independent statements (and templates), which refer to each other. This use of independent templates requires that all labels (circuitGroup, namePkg, name) are globally unique. If, alternatively, the Attribute Class were defined in-line within the Object Class template, e.g.

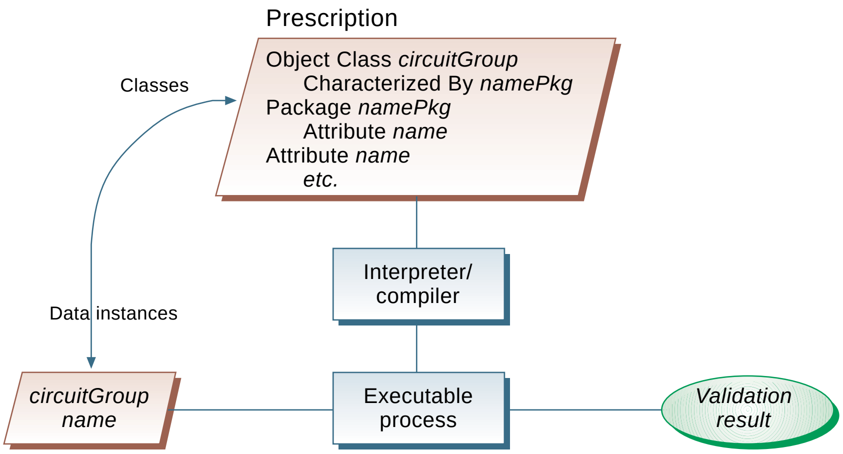


Figure 5 Non-homomorphic classes. The example depicts lack of compliance between the structure of data instances (e.g. circuitGroup containing name) and the corresponding specification containing prefix keywords (e.g. Object Class), extra constructs (e.g. Package), (lack of) indentations (e.g. of Attribute), and maybe word orders (e.g. Object Class statements, Package statements and Attribute statements in different sections)

circuitGroup Object Class
name Attribute Class,

then the Attribute Class label, e.g. name, could have been reused with a different meaning and different value set within a different Object Class label. This could provide a means to define name scopes within superior labels, and the prescription could get a structure identical to that of its instances.

Note that name scoping in Predicate calculus is different from what is exemplified above; constants are globally unique, while variables are unique within the scope of their quantifier. Predicate calculus provides no means of defining terms (constants and variables) within the scope of superior terms. Also, Predicate calculus provides no means of instantiation. However, it provides means to validate non-consistency – of p and ¬p. Note that two unrelated statements p and q are consistent. This is different from instantiation, as well, as instances are only permissible if they are instances of some explicitly defined class.

Principles

The following principles apply to the correspondence between data instances and data classes as seen at the human-computer interfaces:

1 End users need access to data classes to validate their form prior to implementation, for end user help and as an access means to data instances.

2 Data subordination (to other data, as indicated by indentation in the above examples) should be used to define name spaces of data classes and data values; hence, (the end user form of) data are organised into data trees, where each data item has a superior data item and can have several subordinate data items – this applies to both instances and classes.

3 The syntactical form of data classes should be homomorphic to that of their instances, i.e. several instances can be of the same class, and if instance b is subordinate to instance a, then the class of b must be subordinate to the class of a, and if instance b has a reference c to a, then the class of b must have a reference c to the class of a; the homorphism requirement implies that class labels are copied into instances and appear in every instance, and implies that significant duplicates are frequently used.

4 The root node of a data instance tree is referred to as a population relative to the corresponding root node of its data class tree. This root node is referred to as a schema relative to a corresponding root node of a data instance tree. A schema may have several populations, and a population may have several schemata – even if this is not typical.

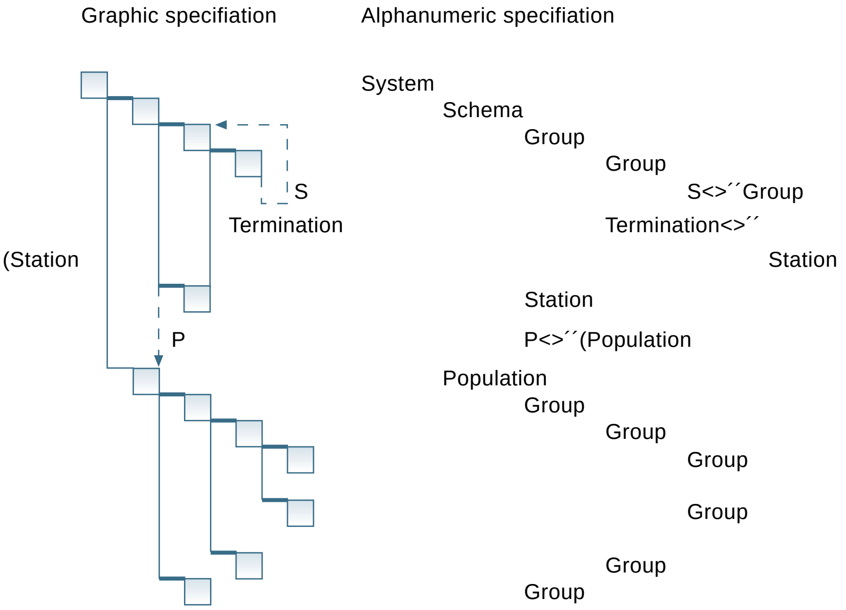


Figure 6 Example system containing one population and its schema; recursion is used to prescribe the Group hierarchy

5 Schema references can be made from class data to other data; hence, end users may access class data and instance data by the same means. This implies that the syntactical form of classes and instances must be identical, and classes may not be marked with labels like Object Class, Attribute Class, etc.

6 A schema reference can be used to define recursion by referring to a superior node in the data class tree.

7 Use of inheritance should be avoided, since this can be provided by the mapping from Application to External schemata, while Schema references, including recursion, and data type references only are used within the prescription, e.g. the Application schema.

Figure 6 exemplifies how data may relate to each other in a data tree.

Rationale

The classification theory of human-computer interaction provides

- direct end user access to and interpretation of specifications without complex transformations and interpretations
- use of local labels in the formal specifications, being typical for headings at the human-computer interfaces
- use of significant duplicates, being typical for graphic symbols at the human-computer interfaces
- definition of schema and population data relative to each other, providing equal access to population and schema data
- avoidance of use of inheritance and interpretation of its implications.

References

1 Meisingset, A. A data flow approach to interoperability. *Telektronikk*, 89 (2/3), 52–59, 1993.

2 ITU-T. *Data oriented human-machine interface specification technique : scope, approach and reference model.* Geneva, ITU, 1993. (ITU-T Recommendation Z.352, 03/93.)

3 ITU-T. *Draft Recommendation Z.35x and Appendices to Draft Recommendations.* Geneva, ITU, 1992. (CCITT COM X-R 12.)

4 ITU-T. *Draft Answers to Q1, 2 and 3/10.* http://www.itu.ch/Standardization, SG10, Reports/. Also published in: Meisingset, A. *The HMI specification technique.* Kjeller, Telenor R&D, 1996. (Report N 54/96.)

Arve Meisingset is Senior Research Scientist at Telenor R&D. He is currently working on information systems planning, formal aspects of human-computer interfaces and middleware standardisation. He is ITU-T SG10 Vice Chairman and the Telenor ITU-T technical co-ordinator.

e-mail:
arve.meisingset@fou.telenor.no

Alliance Approach to the TMN X Interface

RONALD BONICA

Telecommunication service providers maintain alliances through which they resell each other's services. By leveraging the alliance's reach and diversity, each alliance partner offers a complete portfolio of global services to its customers.

Alliance success depends upon inter-operability among alliance partners. Therefore, the alliance must maintain a discipline through which alliance partners exchange orders, trouble reports, performance reports and usage reports. To this end, the International Telecommunications Union specifies the Telecommunications Management Network (TMN) X Interface.

This paper presents an alliance approach to the TMN X Interface. Two principles guide the alliance approach. First, the X Interface must not compromise alliance partner autonomy. Second, the X Interface must facilitate inter-operability. Interface specifications must be publicly available and the interface must not be so complex as to preclude any organization from joining the alliance.

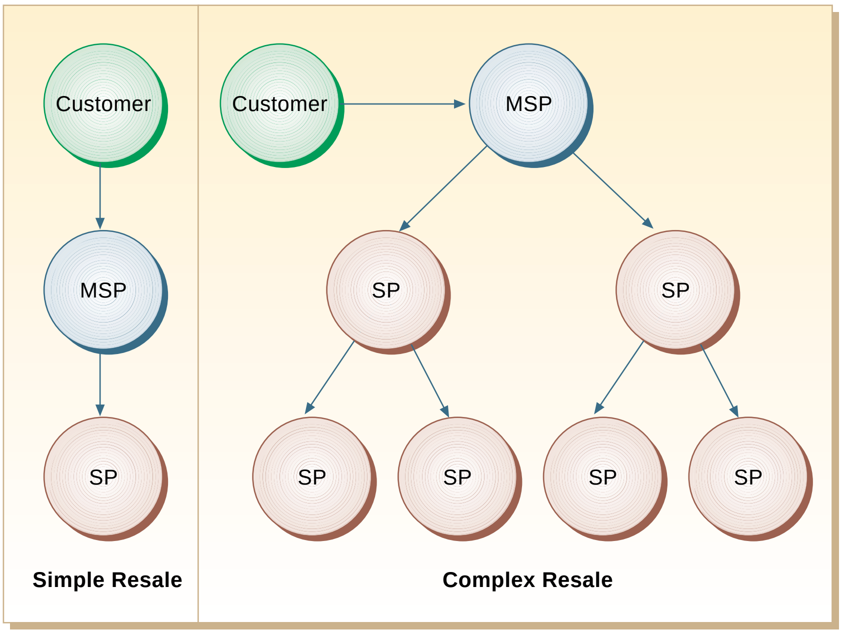


Figure 1 Simple Resale versus Complex Resale

Introduction

Market demands motivate telecommunications service providers (SPs) to resell each other's services. The resale agreement can include two or more SPs. In the simplest case, a customer purchases service from a main service provider (MSP). As the MSP cannot support the service using its own network resources, it establishes a separate service agreement with another SP. The MSP purchases service from the SP and resells the service to its customer.

In the most complex case, a customer purchases service from an MSP. The MSP divides the service into components, determining that it can provide selected components while it cannot provide others. The MSP provides selected service components and purchases the remaining components from one or more SPs.

Each SP, in turn, divides the service request that it receives into components. The SP can provide selected components while it cannot provide others. Therefore, the SP provides selected service components and purchases remaining components from peer SPs.

Figure 1 illustrates both simple and complex resale.

In order to maintain the business arrangement described above, SPs must support a suite of SP-to-SP protocols. ITU-T Recommendation M.3010 [1] identifies the SP-to-SP protocol suite as the X Interface.

This paper explores technical requirements for the X Interface. It extends the X Interface framework presented in ITU-T Recommendation M.3320 [2].

The following sections provide independent views of the X Interface. The first section defines business requirements. The second presents inter-operability requirements in terms of the Telecommunications Management Network (TMN). The third section identifies impediments to TMN development and the final section presents a pragmatic, alliance approach to the TMN X Interface.

Business Requirements

Seamless Service

The MSP must present a seamless service to the customer. In a seamless service, the customer maintains a single service agreement with the MSP and the MSP manages service agreements with SPs on the customer's behalf. The customer can, but need not, be aware that SPs contribute to the service.

Typically, service agreements include the following:

- Service activation scheduling requirements
- Trouble reporting and trouble resolution requirements
- Service availability requirements
- Quality of service requirements
- Performance reporting requirements
- Billing and pricing requirements.

Therefore, the X Interface must support automated exchange of the following information between SPs:

- Order entry and order tracking information
- Trouble reporting and trouble management information
- Performance information (quality of service and service availability)
- Billing and usage information.

The X Interface may be extended to support the real-time exchange of service effecting network events.

Autonomy

Each SP is an autonomous fiscal entity. Therefore, each SP retains the right to manage its own resources. For example, when an SP receives a request for service, that SP determines which resources will be assigned to the service and when those resources will be deployed. The requesting organization cannot directly allocate resources owned by the service providing organization.

As each SP is an autonomous fiscal entity, each SP retains the following rights:

- The right to maintain its own resources. Resources include network resources and operational support systems.
- The right to operate its support systems according to local policy. For example, each SP determines the hours during which its operational supports systems are available and the hours during which they are off-line for preventive maintenance.
- The right to maintain its own data base of record. The data base of record describes all services that the SP has requested through the X Interface or provided in response to requests received through the X Interface.

  Although the SP acknowledges the existence of another data base at the remote end of the X Interface, the SP views its own data base as the "data base of record". SPs need not trust each other to maintain a data base of record.
- The right to specify the technology that supports its enterprise. Therefore, the X Interface must specify a suite of protocols, not a suite of products.

Furthermore, the X Interface must be simple. Integration with SP operational support systems must not be time consuming or expensive. The X Interface must employ only the most commonly available technologies. SPs should not be required to embrace new, complex or exotic technologies in order to partake in the X Interface.

Security

The X Interface must provide the following security features:

- authentication
- non-repudiation
- privacy.

Authentication assures a message receiver of the message originator identity. It protects both the originator and the receiver from malicious parties that masquerade as the message originator.

Non-repudiation assures the message receiver that the message originator cannot deny having sent the message. Privacy assures both the message originator and the message receiver that unauthorized third parties cannot access message contents.

TMN Requirements

TMN Overview

Recommendation M.3010 presents general architectural requirements for a Telecommunications Management Network (TMN). Figure 2 introduces TMN functional blocks and reference points.

M.3010 defines the following functional blocks:

- Work Station Function (WSF) – "The WSF provides the means to interpret TMN information for the management information user."
- Operations System Function (OSF) – "The OSF processes information related to the telecommunications management for the purpose of monitoring/coordinating and/or controlling telecommunications functions including the management function."
- Network Element Function (NEF) – "The NEF is a functional block which communicates with the TMN for the purpose of being monitored and/or controlled."
- Q Adaptor Function (QAF) – "The QAF block is used to connect as part of the TMN those non-TMN entities which are NEF-like and OSF-like."
- Mediation Function (MF) – "The MF block acts on information passing from the OSF to the NEF (or QAF) to ensure that the information conforms to the expectations of the function blocks attached to the MF."

M.3010 also defines the *q*, *x*, *f*, *m*, and *g* reference points. A similarly named interface (*Q*, *X*, *F*, *M* and *G*) services each reference point. The *q* and *x* reference points are relevant to this paper.

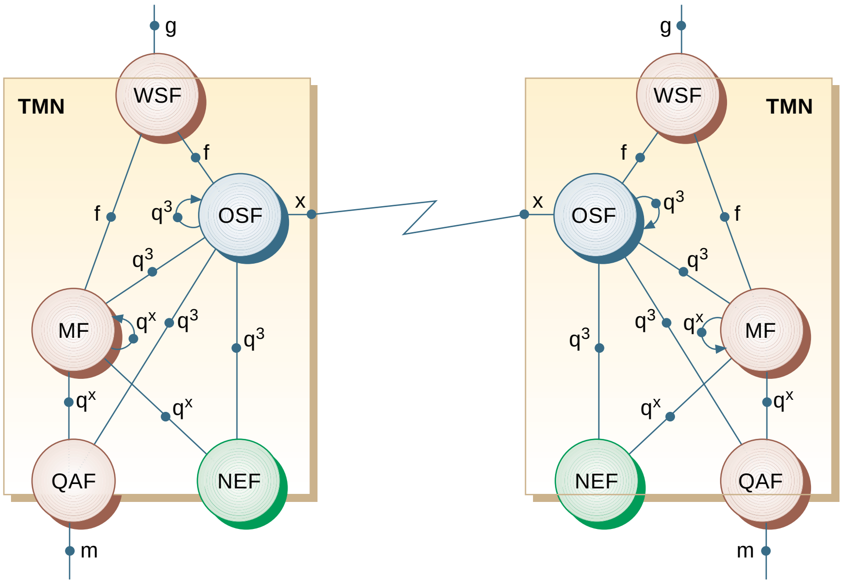


Figure 2 Illustration of Reference Points Between Management Function Blocks From CCITT M.3010, Figure 5/M.3010

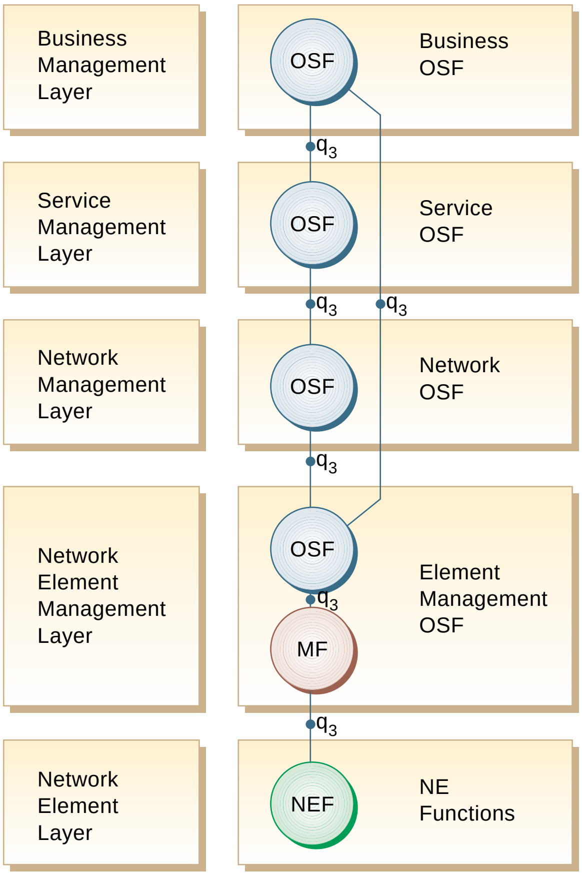


Figure 3 Example of a TMN OS functional hierarchy From M.3010 Figure II-1/M.3010

M.3010 defines the *q* reference point as follows:

> *The* q *reference points serve to delineate a logical part of the information exchange between function blocks as defined by the information model mutually supported by those functions.*

M.3010 identifies two *q* reference point subclasses. These are the q_3 subclass and the q_x subclass.

By contrast, M.3010 defines the *x* reference point as follows:

> *The* x *reference points are located between the OSF function blocks in different TMNs. Entities located beyond the* x *reference point may be part of an actual TMN (OSF) or part of a non-TMN environment (OSF-like). This classification is not visible at the* x *reference point.*

Although M.3010 states that “the protocol definition should seek to minimize the difference between TMN interfaces”, it also recognizes that the Q and X Interfaces are not identical. The X Interface must support the above stated business requirements, particularly those requirements that address autonomy and security.

In order to support above stated requirements, the TMN framework specifies that the Q and X Interfaces typically operate upon unique subsets of the management information model. Furthermore, the TMN framework supports multiple communications services across both the Q and X Interfaces.

Management Information Domains

The Q and X Interfaces typically operate upon unique subsets of the TMN information model. The TMN framework divides OSFs and the information model elements into the following layers:

- Element management layer (EML) – “The EML manages each network element on an individual basis and supports an abstraction of the functions provided by the NE (network element) layer.”[1]
- Network management layer (NML) – “The NML has the responsibility for the management of all network elements, as presented by the EML, both individually and as a set. It is not concerned with how a particular element provides services internally.”
- Service management layer (SML) – “The service management layer is concerned with, and responsible for, the contractual aspects of aspect of the services that are being provided to customers or are available to potential new customers.”
- Business management layer (BML) – “The business management layer has the responsibility for the total enterprise and is the layer at which agreements between operators are made. This layer normally carries out goal setting tasks rather than goal achievement, but can become the focal point for action in cases where executive action is called for.”

Figure 3 illustrates layered OSFs communicating through the Q Interface. In the figure, the Business OSF causes the Service OSF to create, modify and destroy service layer objects by sending messages across the Q Interface. The Business OSF also causes the Element Management OSF to create, modify and destroy element management layer objects by sending messages across the Q Interface.

By contrast, the X Interface is typically restricted to communication between service layer OSFs. Using the X Interface, an MSP causes an SP to manipulate service layer objects within the SP domain. The SP service layer OSF causes the SP network layer OSF to manipulate SP network layer objects by sending messages through the Q Interface.

[1] *As the network element layer resides outside of the OSF domain, it is not included in this list.*

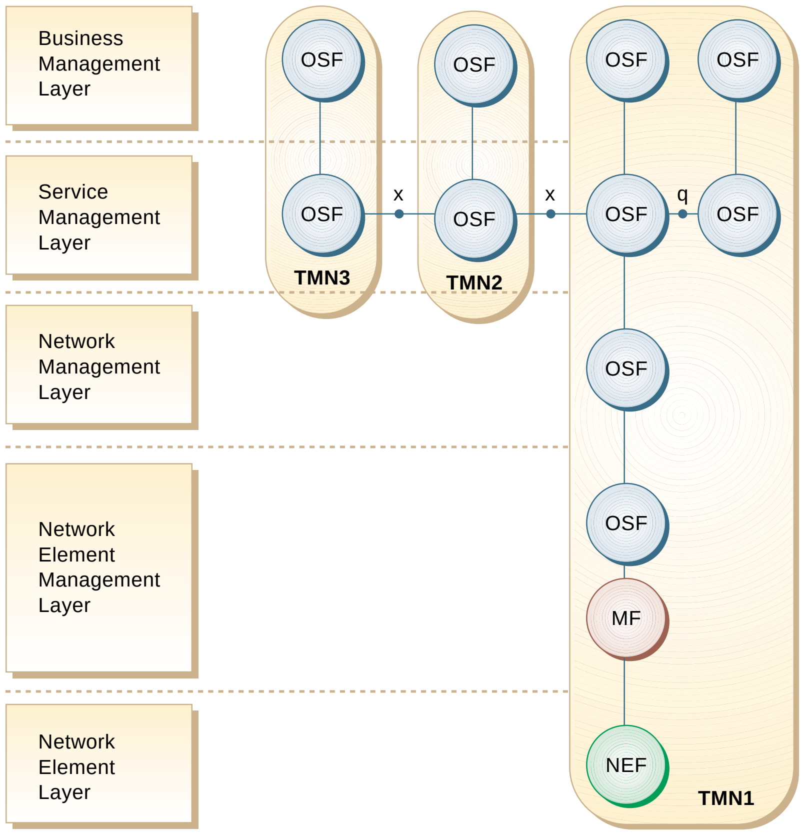


Figure 4 Examples of Value Added Services From M.3010 Figure II-2/M.3010

Therefore, the MSP cannot determine when or how the SP realizes the service in terms of network layer objects. The SP retains functional autonomy.

Restricting the X Interface to service layer interactions insulates the MSP from SP network and network element layer details. Technical autonomy is enhanced in that the MSP and SP can model the network and network element layers differently, without affecting their ability to interact at the contractual, service layer. Figure 4 illustrates SPs (in this case, value added service providers) interacting at the service layer.

The TMN framework permits the service layer of one TMN to communicate with a lower layer of another TMN when the problem domain permits. For example, the service layer of one TMN may communicate with the network layer of another TMN when functional autonomy between TMNs is not required and when it is not economical to build a service layer OSF in the second TMN.

M.3010 warns that "implementation of such an architecture may require very strong access control mechanisms." Figure 5 illustrates inter-layer communication across the X Interface.

Communication Services

Recommendation M.3320 states that the following communication services are available to the TMN X Interface:

- Interactive Services
- File Transfer Services
- Directory Services
- Store and Forward Services.

Interactive Services

Using interactive services, SPs access objects that reside within each other's domains. As the name implies, interactive services support near-real-time access. In order to support interactive services, SPs maintain a manager/agent relationship.

Figure 6 illustrates the manager/agent relationship. In the figure, the managed system maintains managed objects and agent software. Managed objects provide an abstracted view of a managed resource. For example, if the managed resource is a frame relay switch, managed objects might represent frame relay permanent virtual circuits. Agent software accesses managed resources in order to maintain managed objects.

Agent software represents managed objects through a hierarchic data structure called the Management Information Tree (MIT). The manager and agent must agree upon the structure of managed objects before communicating through the interactive service. An ASN.1 Management Information Base (MIB) definition formalizes the structure of managed objects.

In order to access managed objects, the manager sends messages to the agent. The agent, in turn, performs management operations on the manager's behalf.

According to Recommendation Q.812 [3], the Common Management Information Service Element (CMISE) [4] provides interactive services for the TMN X Interface. The following is a list of CMISE primitives:

- M-CREATE – create a managed object instance
- M-DELETE – destroy one or more managed object instances
- M-GET – retrieve selected attributes from one or more managed object instances
- M-SET – modify selected attributes for one or more managed object instances
- M-ACTION – perform an object specific operation upon one or more managed object instances
- M-CANCEL-GET – cancel a previously issued M_GET operation
- M-EVENT-REPORT – register for notification when one or more managed object instances change.

Most CMISE primitives operate upon one or more managed object instances. In order to specify which instances the primitive operates upon, the manager "scopes" its requests. Scoping relies upon the hierarchic nature of management information.

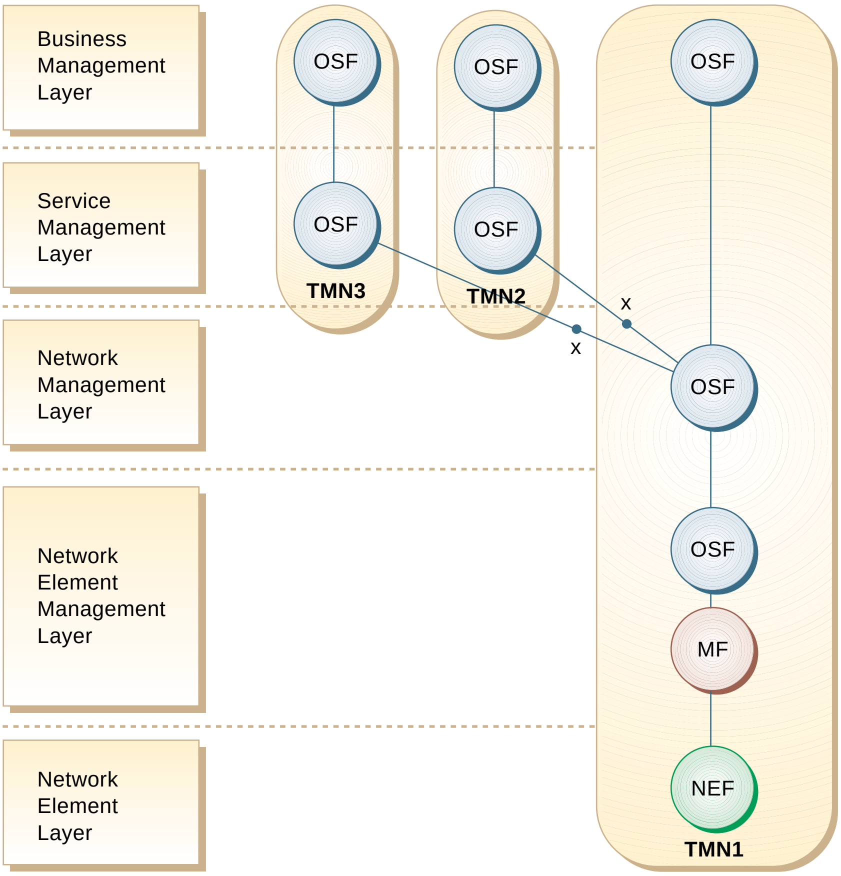


Figure 5 Examples of Inter-TMN OS functional connectivity From M.3010 Figure II-3/M.3010

When the manager issues a request, it specifies a point on the hierarchic MIT, called the base object. It also specifies to which of the following the requested action should be applied:

- the base object only
- the (n)-level subordinated to the base object
- the base object and all of its subordinates down to and including the (n)-level
- the base object and all of its subordinates.

The manager can exempt selected object instances from the operation by "filtering". For example, using a single M_DELETE primitive, a manager can destroy all object instances subordinate to a base object except those with attribute "A" equal to five.

The manager/agent relationship is asymmetric. For example, only the manager can issue an M-CREATE request. In order to support symmetric relationships, both systems (managed and managing) must be equipped with both manager and agent software.

Figure 7 depicts upper layer protocol profile for interactive services.

File Transfer Services

SPs access each others files using the file transfer service. Specifically, SPs execute the following actions upon files that reside in another SP's domains:

- Create/delete file
- Read file attributes (e.g., creation time, last modification time)
- Open/close file
- Read/write contents.

Recommendation Q.812 identifies the file transfer service for the TMN-X Interface as being provided by File Transfer, Access and Management (FTAM) [5]. Specifically, the following FTAM file structures are supported across the X Interface:

- Unstructured text files (FTAM-1 document type)
- Unstructured binary files (FTAM-3 document type)
- Sequentially ordered files (NBS-6 document type).

Because the X Interface supports only the above listed file structures, FTAM random access capabilities are not available at the X reference point. (FTAM random access capabilities apply only to document types FTAM-2 and FTAM-4 documents.)

Figure 8 depicts upper layer protocol profile for file transfer services.

Directory Services

The directory service allows distribution of management information over a potentially unlimited number of OSI end systems. Despite its physical distribution, the end user or application invoking the directory service perceives management information as a single physical unit.

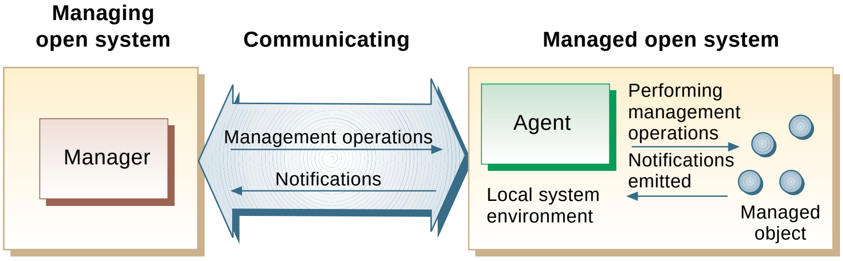


Figure 6 Interaction between Manager, Agent and objects From M.3010 Figure 8/M.3010

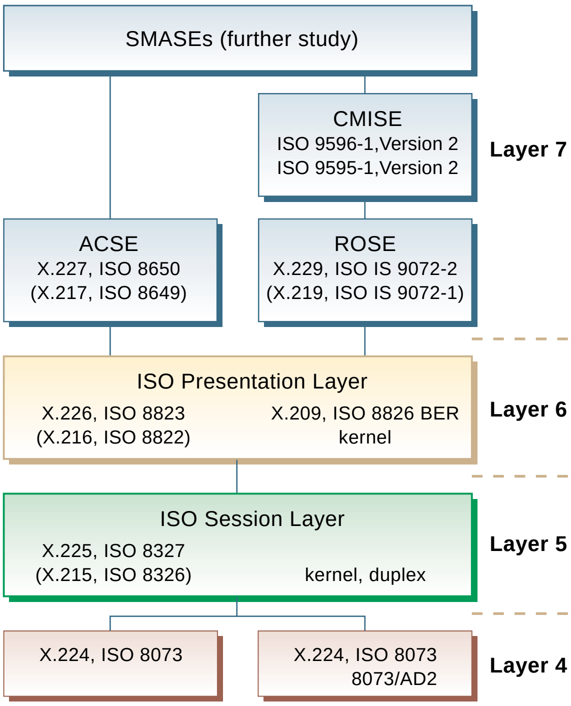


Figure 7 Protocol profile for network management which uses transaction function
From Q.812, Figure 1/Q.812

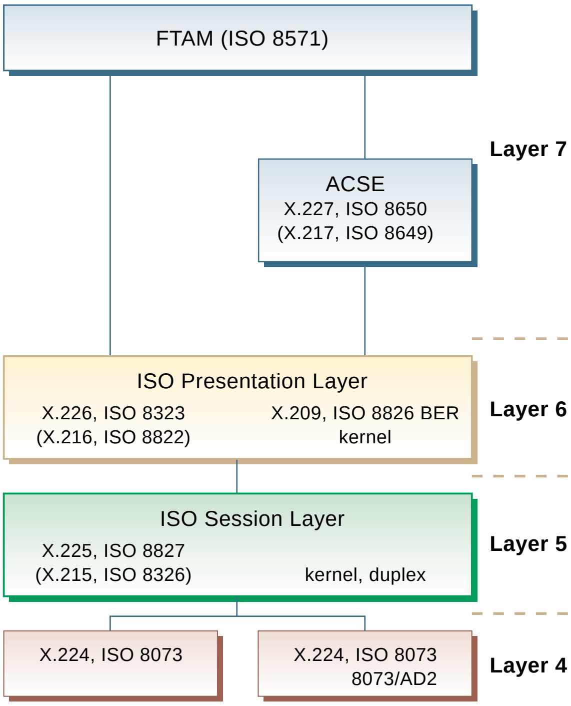


Figure 8 Protocol profile for network management which uses file transfer
From Q.812, Figure 2/Q.812

Recommendation Q.812 defines the directory services for the TMN-X Interface as being provided by the X.500 series of recommendations.

Store and Forward Services

Although Recommendation M.3320 states that store and forward services are available to the X Interface, it provides no specifics. Therefore, the current document provides a generic description of store and forward services.

When an application submits a protocol data unit (PDU) to the store and forward service, the store and forward service executes the following actions:

- Assign a sequence number to the PDU.
- Commit the PDU and sequence number to persistent storage
- Return the sequence number to the application.

When the application receives the sequence number, it is certain that the PDU will be delivered to the remote application as soon as possible. It is also certain that PDUs will be delivered to the remote application in the same order in which they were submitted to the local store and forward service entity.

Although the application can query the PDU delivery status, it cannot retract the PDU from the store and forward entity. The following is a list of PDU delivery status:

- Pending transmission – the PDU is awaiting transmission from the local store and forward entity to the remote store and forward entity.
- Pending delivery – the PDU has been received and made persistent by the remote store and forward entity. It is awaiting delivery to the remote application.
- Delivered – the PDU has been delivered to the remote application.

Impediments to a TMN X Interface

The communication services described above are feasible if all SPs maintain a data base of record that is organized according to TMN sanctioned information models. In this case, the mapping between data base of record and management information tree is trivial. The

CMISE agent simply provides remote access.

Unfortunately, many SPs do not maintain a data base of record that is organized according to TMN sanctioned information models. The following are common causes of deviation:

- Many SPs maintain legacy systems that pre-date TMN sanctioned information models.
- Market pressures motivate SPs to provide services before standards bodies codify a corresponding information model.
- Support systems distinguish themselves from one another through unique data base organizations.

Therefore, the CMISE agent must implement complex adaptation functions in order to compensate for the “impedance mismatch” between data base of record and management information tree. Typically, adaptation function development is very costly and adaptation software is not reusable.

The impedance mismatch problem is well known to CMISE application developers. Because of the impedance mismatch problem, many CMISE agents do not implement scoping, filtering, or event forwarding registration.

An Alliance Approach

Having examined SP business requirements and TMN inter-operability requirements, we can formulate a pragmatic, alliance approach to the TMN X Interface. The alliance approach extends, but does not deviate from, the canonical TMN X Interface described above.

The alliance approach contains information, management protocol and transport aspects. Information aspects address which objects are to be shared by SPs through the X Interface. Management protocol aspects address how SPs coordinate management functions (e.g., order management, trouble management). Communication aspects address how SPs communicate with each other.

Information Aspects

Objects shared through the X Interface must be sufficiently well defined that SPs cannot misinterpret their meaning. In order to achieve the required level of specificity, each object definition must include the following:

- A concise textual description
- A complete list of attributes, with a concise textual definition for each attribute
- A complete list of methods, with a concise textual definition for each method.

Concise object modeling is a monumental task. As object model size increases, so do the following:

- the probability that the object model contains one or more imprecisely defined objects
- the probability that the object model is internally inconsistent
- the probability that some aspect of the object model will be misunderstood by an SP.

Therefore, the object model shared through the X Interface must include as few objects as possible. In order to restrict model size, as well as preserve SP autonomy, shared objects must be restricted to the TMN service layer.

Each object shared across the TMN X Interface is categorized as either service specific or service independent. Service specific objects define the services that SP purchase from one another. They contain contractual attributes, configurable attributes, performance attributes and usage attributes.

Contractual attributes define service type, service availability dates, service availability requirements (e.g., mean time between failures) and quality of service requirements. Each contractual attribute contributes to the service price. Configurable attributes specify additional service definition parameters that do not contribute to service price.

Performance attributes define metrics for availability and quality of service. Usage attributes define metrics for usage based billing.

Each time the SP community agrees to support a new service through the TMN X Interface, it must define at least one new service specific object. The SP community should seek object modeling guidance from the most widely accepted standards. For example, the objects that describe Internet service should resemble MIB-II [6].

Service independent objects represent generic operations between SPs. Service independent objects include the following:

- Order
- Trouble report
- Performance report
- Usage report.

Service independent objects should be defined by international standards whenever possible. For example, ITU-T Recommendation X.790 [7] defines the trouble report and several associated objects (e.g., contact).

If widely accepted international standards are not available, the SP community should define service independent objects with guidance from emerging standards. For example, the Open Service Layer Protocol [8] draws upon ITU-T Draft Recommendation M.3208.1 [9] and emerging Network Management Forum proposals in order to define the order object.

Above all, service independent objects must be simple. They must support simple management protocols.

Management Protocol Aspects

The X Interface must support single state and multi-state protocols. Single state protocols support automated exchange of performance and usage information. Single state protocol supports the following scenario:

- The SP creates a service independent object within its own data base of record. The service independent object represents a performance report or usage report.

  The SP also creates zero or more service dependent service objects. It associates the service independent objects with the service dependent object.
- The SP sends a message to the MSP informing the MSP that the new objects have been created.
- The MSP creates identical objects within its data base of record.
- The MSP sends a confirmation message to the SP.

In a single state protocol, the SP can send messages to the MSP in batch mode. In this case, the MSP can confirm reception of the entire batch with a single message.

Multi-state protocols support the following scenario:

- The MSP creates a service independent object within its own data base of record. The service independent object represents an order or trouble report.

  The MSP also creates zero or more service dependent objects. It associates the service independent objects with the service dependent object.
- The MSP sends a message to the SP informing the SP that the new objects have been created.
- The SP creates identical objects within its data base of record.
- The SP processes the objects contained by its data base of record according to local policy. It updates object status and engineering details as appropriate.
- Each time the SP updates the objects contained by its data base of record, it sends a status update message to the MSP. Each status update message contains a status indication and all engineering detail specified by the SP.
- The MSP receives the status update message and updates the corresponding objects in its data base of record.
- When the MSP receives a status update message that indicates the order has been completed or the trouble ticket has been closed, the MSP sends the SP a message indicating whether or not it is satisfied with how the management operation has been executed.

Once the MSP sends its initial message to the SP, the MSP's ability to communicate with the object contained by the SP data base of record is extremely limited. Specifically, the MSP can

- monitor status update messages sent from the SP
- synchronize its copy of the data base object with the SP copy by requesting a retransmission of the last status update message
- send a message to the SP that appends free form text to the object. The message can also escalate an issue or defer a trouble ticket.

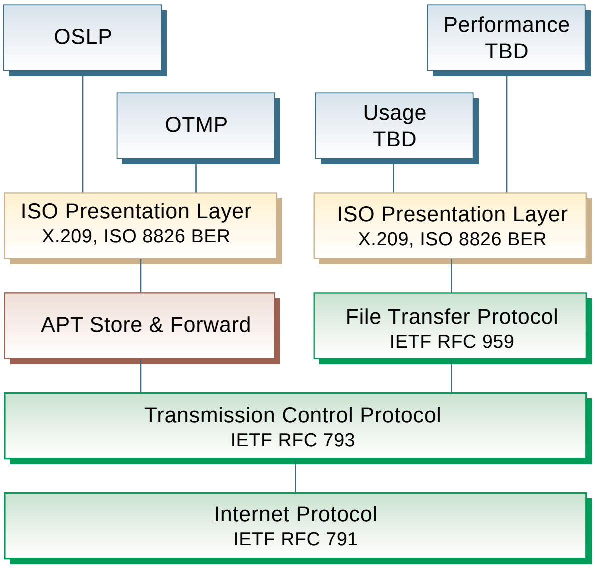

OSLP
OTMP
Performance TBD
Usage TBD
ISO Presentation Layer X.209, ISO 8826 BER
ISO Presentation Layer X.209, ISO 8826 BER
APT Store & Forward
File Transfer Protocol IETF RFC 959
Transmission Control Protocol IETF RFC 793
Internet Protocol IETF RFC 791

Figure 9 Unified View of the Alliance Approach to the TMN X Interface

In order to modify or cancel a management operation (i.e., order or trouble ticket), the MSP must create a new object within its data base of record. The new object supersedes or cancels the original object. The MSP sends a message to the SP and the SP creates an identical object within its data base of record. The SP processes both objects according to the scenario described above.

Multi-state protocols minimized the impedance mismatch effect that is characteristic of CMISE. SPs execute only a few course-grained operations upon objects that reside in each others domain. Therefore, SPs need not implement complex adaptation functions that map every possible view of their management information tree to their data base of record. SPs implement only a small set of adaptation functions that support object creation, status update reporting, and operation closure.

The Open Service Layer Protocol (OSLP) is a multi-state protocol that supports ordering. The Open Trouble Management Protocol (OTMP) is a similar multi-state protocol that supports trouble management. OSLP is presented in another paper [8] in this issue of *Telektronikk*, while OTMP is not yet published.

Communication Aspects

The Basic Encoding Rules (BER) described in ITU-T Recommendation X.209 [10] provide OSI presentation layer services to both single state and multi-state protocols.

All currently defined single state protocols send BER encoded messages through the file transport service while all currently defined multi-state protocols send BER encoded messages through the store and forward service. In the future, however, new single state protocols that employ store and forward services may be developed. Similarly, new multi-state protocols that employ file transfer services may be developed.

The File Transfer Protocol (FTP) described in IETF RFC 959 [11] provides file transfer services to upper layer applications. This augmentation to the communication services described in Q.812 is justified by the following arguments:

- FTP provides services commensurate with those provided by the FTAM subset profiled in Q.812.
- FTP is available in the public domain.
- FTP has gained much wider commercial acceptance than FTAM.

Store and forward services are provided by the Alliance Point-to-Point Transport Protocol (APT). APT is defined in the OSLP documentation suite.

Lower layer services are provided by the Transmission Control Protocol (TCP) [12] and the Internet Protocol (IP) [13].

As stated in Recommendation M.3320, SPs maintain pair-wise agreements regarding encryption and security administration. Because each SP is constrained by its national security and encryption policy, it may not be possible to specify a single, global security policy.

Figure 9 depicts a unified view of the alliance approach to the TMN X Interface.

Conclusions

The X Interface approach proposed herein is an expedient solution. Although it is less robust than the canonical implementation described in Recommendation M.3320, it permits SPs to reap benefits from the TMN architecture in the short term.

References

1 ITU-T. *Principles for a Telecommunications Management Network.* Geneva, ITU, 1992. (ITU-T Recommendation M.3010.)

2 ITU-T. *Management Requirements Framework for the TMN X Interface.* Geneva, ITU, 1997. (ITU-T Recommendation M.3320.)

3 ITU-T. *Upper Layer Protocol Profiles for the Q3 and X Interface.* Geneva, ITU, 1996. (ITU-T Recommendation Q.812.)

4 ISO. *Information Processing Systems : Opens Systems Interconnection : Common Management Information Protocol Specification (CMIP).* Geneva, 1995. (ISO 9595-1 version 2.)

5 ISO. *Information Processing Systems : Open Systems Interconnection : File Transfer, Access and Management, Part 1: General Introduction.* Geneva. (ISO 8571-1.)

6 IETF. *Management Information Base for Network Management of TCP/IP-based internets.* MIB-II, 3/91. (RFC 1213.)

7 ITU-T. *Trouble Management Function for ITU-T Applications.* Geneva, ITU, 1995. (ITU Recommendation X.790.)

8 Bonica, R P. Open Service Layer Protocol (OSLP). *Telektronikk*, 94 (1), 48–54, 1998 (this issue).

9 ITU-T. *TMN Management Services for Dedicated and Reconfigurable Circuits Network : Leased Circuit Services.* Geneva, ITU, 1997. (ITU-T Draft Recommendation M.3208.1.)

10 ITU-T. *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).* Geneva, ITU, 1988. (ITU-T Recommendation X.209.)

11 IETF. *File Transfer Protocol (FTP).* 1985. (RFC 959.)

12 IETF. *Transmission Control Protocol.* 1981. (RFC 793.)

13 IETF. *Internet Protocol.* 1981. (RFC 791.)

Ronald P. Bonica is a member of the MCI Internet Engineering Team. He is currently under contract to the Concert Frame Relay Team and participating in the Concert Alliance Interface Standards effort. His interests are TMN and Internet architecture.

e-mail:
rbonica@mci.net

The Open Service Layer Protocol (OSLP)

RONALD BONICA

Market demands motivate telecommunications service providers to resell each other's services. In order to manage the resale environment, service providers must agree upon a suite of protocols through which they can exchange management information. The protocol suite must support order management, trouble management, performance reporting and usage reporting.

The Telecommunications Management Network (TMN) X Interface addresses the required protocol suite in general terms. Another article in this issue of *Telektronikk* proposes "an alliance approach" to the protocol suite. This paper presents the Open Service Layer Protocol (OSLP), an order management protocol that applies the alliance approach to the TMN X Interface.

OSLP services are organized into layers. The lower layer provides a stateless protocol through which service providers exchange order information. The upper layer is a significant extension to the TMN X interface in that it maintains persistent information concerning the state of subcontracted orders.

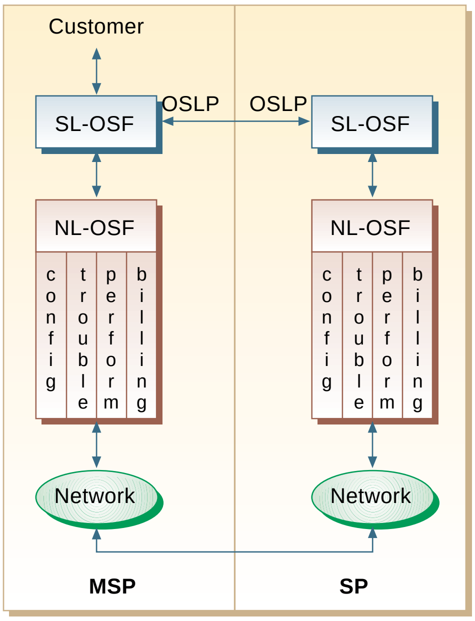


Figure 1 The MSP/SP Relationship

Introduction

OSLP supports the automated exchange of ordering information across Service Provider (SP) boundaries. It complies with the Alliance Approach to the TMN X Interface [1].

OSLP maximizes inter-operability and autonomy among SPs. It maximizes inter-operability by specifying a concise service order syntax. The concise service order syntax includes a service independent component and many service specific components. The service independent component codifies a generic order object. The generic order object specifies all order attributes and the states through which each order must progress en route to completion. The service independent component also specifies several generic objects that are associated with orders (i.e., contact, location).

A service specific component describes each service that SPs offer to one another. For example, one service specific component describes the frame relay service, while another describes the IP service. Service specific components include all service configuration parameters (e.g., bandwidth, quality of service).

OSLP contributes to SP autonomy by maintaining an environment in which each SP retains the following rights:

- The right to manage its own network resources. Although SPs request services using OSLP, they cannot dictate which network resources will support the service or when those resources will be deployed.
- The right to operate its support systems according to local policy. For example, each SP determines the hours during which its operational supports systems are available and the hours during which they are off-line. OSLP adapts by sending each message through a point-to-point store and forward service.
- The right to security. OSLP messages are encrypted to ensure authentication, non-repudiation and privacy.
- The right to maintain its own data base of record. An SP's data base of record describes all services that the SP has requested via OSLP or provided in response to OLSP requests.

  Although the SP acknowledges the existence of another data base at the remote end of the OSLP interface, the SP views its own data base as the "data base of record". SPs need not trust each other to maintain an accurate data base of record.
- The right to progress orders as per local policy. Although OSLP defines a generic order object and the states through which orders progress en route to completion, it does not specify which activities occur in each state.
- The right to specify the technology that supports its enterprise. As OSLP restricts itself to communication regarding TMN service layer objects, SPs are free to model business, network and network element layer objects as per local policy.

  Furthermore, OSLP relies upon only the most widely accepted standards (e.g., TCP, IP) for lower layer services. SPs need not embrace new, costly or exotic technologies in order to implement OSLP.

The following sections describe significant aspects of OSLP.

The OSLP Business Relationship

OSLP formalizes the business relationship depicted in Figure 1. In the figure, the customer contracts with a main service provider (MSP) for service.

The MSP submits an order to its local service layer operation systems function (SL-OSF). The MSP's SL-OSF divides the order into service components and identifies components that the MSP can provide using its own network resources. The MSP network might provide all components, selected components, or no components.

The MSP's SL-OSF sends a request for locally provided components to a local network layer operation systems function (NL-OSF). The MSP's NL-OSF provisions and manages the locally provided service components as per local policy.

The MSP's SL-OSF also identifies service components that the MSP cannot provide using its own network resources. Using OSLP, the MSP orders those services from a peer service provider (SP).

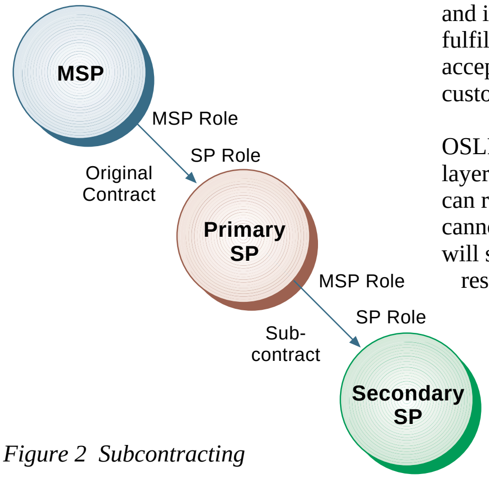


Figure 2 Subcontracting

When the SP SL-OSF receives its OSLP order, it determines whether its local network can fulfil the entire request. If so, the SP SL-OSF sends a message to its local NL-OSF. The SP's NL-OSF provisions the order as per local policy.

When the SP's NL-OSF fulfils the request, it sends a completion message to its local SL-OSF. The SP SL-OSF receives the completion message and sends an OSLP completion message to the MSP's SL-OSF. The MSP accepts the services and the SP bills the MSP for services rendered.

When the MSP SL-OSF receives completion messages from the SP's SL-OSF and its local NL-OSF, it reports service fulfilment to the customer. The customer accepts the service and the MSP bills the customer.

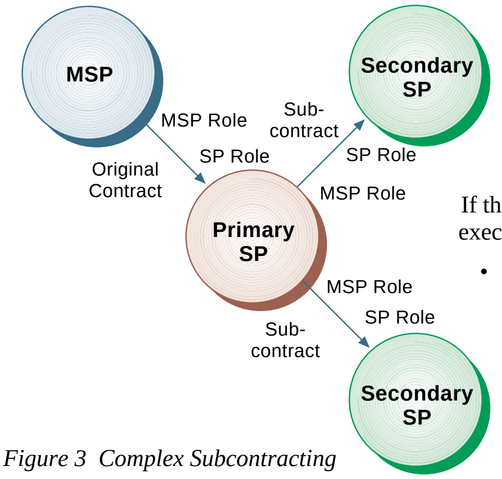


Figure 3 Complex Subcontracting

OSLP messages contain TMN service layer objects only. Therefore, the MSP can request services using OSLP, but cannot dictate which network resources will support the service or when those resources will be deployed.

OSLP messages describe service connections and service endpoints. A service endpoint is an interface through which a party outside of the SP network interface interfaces with the SP network. User-to-network interfaces implement an endpoint through which users interface with the SP network. Network-to-network interfaces implement an endpoint through which peer networks interface with the SP network.

OSLP messages never describe how an SP connects endpoints within its own network, because TMN assigns such details to the network layer. OSLP messages never include internal implementation or routing details.

OSLP codifies only those messages that pass between the MSP's SL-OSF and the SP's SL-OSF. Organizations can implement their internal service layer to network layer interface using either proprietary protocols, the canonical Q3 Interface [2] or an OSLP variant that addresses network layer objects.

Subcontracting Relationships

SPs receive requests for services that they cannot provide using their own network resources. When an SP receives such a request, it can either reject or subcontract the request. If the SP rejects the request, it sends an OSLP message to the MSP indicating order rejection.

If the SP subcontracts the request, it executes the following procedure:

- Divide the request into components
    - Identify service components that the local network can support
- Send a request for locally supported components to the local NL-OSF
- Identify a secondary SP for the remaining service components
- Send an OSLP service request to the secondary SP
- Send the MSP an OSLP message informing it of the subcontracting arrangement
- Receive an OSLP completion message from the secondary SP
- Send an OSLP acceptance message to the secondary SP
- Receive a completion message from the local NL-OSF
- Send an OSLP completion message to the MSP.

In short, the primary SP maintains an original contract with the MSP and a subcontract with the secondary SP. The primary SP assumes the SP role with respect to the original contract and the MSP role with respect to the subcontract.

Figure 2 depicts the subcontracting relationship.

Complex and Recursive Subcontracting

In the example above, the primary SP requests services from a single secondary SP. In complex cases, the primary SP divides the original request into many parts and requests service components

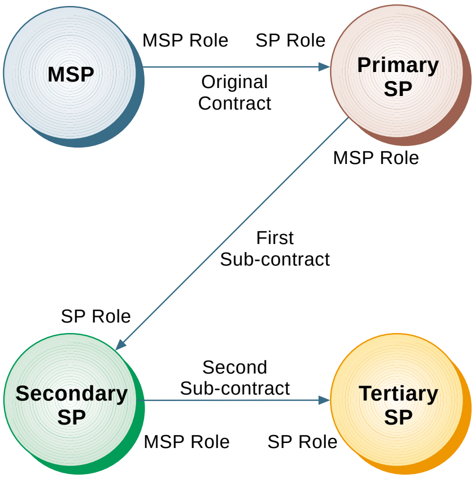


Figure 4 Recursive Subcontracting

from two or more secondary SPs. Figure 3 illustrates complex subcontracting.

Furthermore, the subcontracting relationship is recursive. The secondary SP can subcontract services from yet another SP. Figure 4 illustrates recursive subcontracting.

All subcontracting details are relayed upstream to the MSP.

OSLP Services

OSLP services are organized into the following layers:

- Primitive services (provided by the lower layer)
- Subcontracting services (provided by the upper layer).

Primitive services are stateless. Although MSP and SP applications employ primitive services to exchange order information, the OSLP primitive layer does not maintain a data base of outstanding orders. MSP and SP applications must maintain a data base of outstanding orders and manage these orders as per local policy.

OSLP provides one set of primitive services to MSPs and another set of primitive services to SPs.

Subcontracting services are statefull. When a primary SP divides an incoming order into components and subcontracts each component to a secondary SP, the primary SP's OSLP subcontracting layer records the incoming order and all component orders in a local data base. The primary SP's subcontracting layer also forwards all component orders to the secondary SPs.

When secondary SPs achieve provisioning milestones, they update their order status and send the primary SP an OSLP message notifying it of the update. The primary SP's OSLP subcontracting layer executes the following actions:

- Update the component order status in the local OSLP data base
- Re-evaluate the status of the original order received from the MSP
- Store the original order status in the local OSLP data base
- Send the MSP an OSLP message indicating the original order status and the status of all component orders
- Inform the primary SP's higher level applications of new order status.

The subcontracting service constitutes a significant extension to the canonical TMN X Interface. Specifically, the subcontracting service introduces the concept of a statefull, persistent transaction. Although the subcontracting service is optional, it is useful to SPs that engage in subcontracting.

OSLP MSP Services (Primitive Layer)

OSLP provides the following services to MSPs:

- MSPs create, modify and cancel orders using OSLP.
- MSPs annotate orders with free form text using OSLP.
- MSPs escalate and de-escalate ordering issues using OSLP.
- MSPs query order status using OSLP.
- OSLP notifies the MSP when SPs completes provisioning milestones.
- MSPs accept completed services using OSLP.
- OSLP notifies the MSP when an SP unilaterally re-engineers a service.
- OSLP notifies the MSP when an SP annotates an order with free form text.
- OSLP notifies the MSP when the SP requests de-escalation.

Order Creation

MSPs create the following order types:

- Bid request
- Provisioning request.

SPs respond to a bid request with the following:

- An estimated completion date (mandatory)
- Engineering details (supplied at SP discretion)
- A price (supplied upon MSP request).

SPs respond to a provisioning request by provisioning and testing the requested service. SPs inform the MSP each time a provisioning milestone is completed.

Each order, regardless of type, contains zero[1] or more service components. Each service component contains an action, service independent details, and service specific details.

MSPs request the following service actions:

- Install a new service
- Change an existing service
- Discontinue an existing service.

MSPs specify the following service independent details for each service:

- Service identifiers (e.g., access circuit identifiers)
- Service type (e.g., IP access, X.25 access)
- Contact details
- Billing details.

A single OSLP order can contain components that specify different actions and service types. For example, an MSP can request the following using a single OSLP order:

- Installation of an IP service
- Discontinuation of an X.25 service
- Modification of a frame relay service.

The SP cannot charge for any component of an order until all components of order are complete and accepted by the MSP.

Using OSLP orders, the MSP can queue multiple actions against a single service. For example, on a single day, an MSP can:

- Schedule a circuit for installation in January
- Schedule the same circuit for upgrade in February
- Schedule the circuit for disconnection in March.

Order Modification and Cancellation

MSPs create orders that supersede or cancel other orders. The SP responds twice. First, the SP responds to the initial order, specifying that it is aborted at MSP request. Next, the SP responds to

[1] *A supplier might place an order with no service components in order to update contact information.*

the modifying or canceling order, specifying that it is in progress or complete.

OSLP does not permit MSPs to modify or cancel orders directly. MSPs must issue a new order that supersedes or cancels an existing order.

Order Annotation

MSPs can append annotations (i.e., free form text) to an order at any time. OSLP forwards the annotations to the supplier.

Using annotations, MSPs can escalate ordering issues to the SP's management. MSPs can also de-escalate issues when they have been notified that the issue has been resolved.

OSLP notifies the MSP of annotations generated by the SP. OSLP also notifies the MSP when the SP requests de-escalation.

Order Status Inquiry

MSPs query the status of their orders by sending the SP an OSLP message. SPs respond to the OSLP query by re-evaluating the order's status and sending an OSLP status update message to the MSP. The OSLP status update message includes all service details that were provided by the SP, including the order status.

Milestone Notification / Order Status Update

OSLP notifies MSPs when SPs achieve the following provisioning milestones:

- The SP has *claimed* the order. The SP will fulfil the order without subcontracting.
- The SP has *distributed* the order. The SP will fulfil the order, but has subcontracted some or all of the order to secondary SPs.
- The SP has *subcontracted* the order. The SP will fulfil the order, but has subcontracted some or all of the order to secondary SPs. The secondary SPs (or their SPs) have claimed their portions of the order.
- The SP has *engineered* the order. For example, the SP has assigned access ports.
- The SP has *completed* the order. It has provisioned and tested the service.
- The SP has *failed* the order due to order ambiguity or inability to provide the requested service.
- The SP has *aborted* the order at the MSP's request.

Order Acceptance / Rejection

After OSLP notifies the MSP that the SP has provisioned and tested the service, the MSP must either accept or reject the service within a configurable time period. If the MSP accepts the service or fails to take action within the configurable time period, the SP initiates the billing process.

If the MSP rejects the service, the MSP and SP must negotiate the issue outside of OSLP. Typically, the SP resolves the problem, and the MSP accepts the service at a later date.

Re-engineering Notification

OSLP notifies the MSP when an SP unilaterally re-engineers a service. The SP can re-engineer the service in response to a failure condition or in conjunction with a network grooming effort. The SP also can re-organize support staff, assigning new support contacts to a service.

SP Services (Primitive Layer)

OSLP provides the following services to suppliers:

- OSLP notifies the SP of incoming orders.
- SPs notify the MSP of milestone completion using OSLP.
- OSLP notifies the SP of incoming status inquires.
- SPs respond to status inquires using OSLP.
- OSLP notifies the SP when the MSP accepts service order completion.
- SPs notify the MSP of unilateral service re-engineering using OSLP.
- SPs annotate orders with free form text messages and distribute annotations to MSPs using OSLP.
- SPs request de-escalation of ordering issues using OSLP.
- OSLP notifies the SP when the MSP annotates an order with free form text.
- OSLP notifies the SP when the MSP escalates or de-escalates an ordering issue.

These services are symmetric with those provided to MSPs.

Subcontracting Services (Subcontracting Layer)

The subcontracting service integrates MSP and SP services. The subcontracting service includes:

- Order binding
- Delayed order binding.

The following example illustrates order binding:

> *An MSP orders a frame relay permanent virtual circuit (PVC) from a primary SP. The PVC connects an endpoint in London to an endpoint in Rome. The primary SP cannot fulfil the order using its own network resources. Therefore, it divides the order into two components. One component connects the London endpoint to a frame relay network-to-network interface (NNI) in Milan. The other component connects the Roman endpoint to an NNI that is co-located with the first NNI in Milan.*
>
> *The primary SP assigns one service component to a secondary SP in England and the other to a secondary SP in Italy. Using OSLP subcontracting service, the primary SP creates a new order representing each service component and binds the new orders to the original order from the MSP.*
>
> *The OSLP subcontracting layer records the original order and the new orders in its local data base. It sends one new order to the secondary SP in England and the other to the secondary SP in Italy.*
>
> *As the secondary SPs provision their components, they send status update messages to the primary SP. When the primary SP's subcontracting layer receives these messages, it re-evaluates the status of the original order from the MSP.*
>
> *The primary SP's subcontracting layer composes a status update message that includes information concerning the original order, as well as the two new orders. The primary SP's subcontracting layer sends this message to the MSP as well as to the primary SP's higher lever applications.*

In the example above, the primary SP chooses the NNI at which the two PVC segments meet. Therefore, the primary SP issues complete service orders to both secondary SPs, simultaneously, specifying a network-to-network interface for each.

Business policy can require that one or the other secondary SP specifies the network-to-network interface. Therefore, OSLP supports delayed order binding. The following frame relay example illustrates delayed order binding:

> *An MSP orders a frame relay PVC from a primary SP. The PVC connects an endpoint in London to an endpoint in Rome. The primary SP cannot fulfil the order using its own network resources. Therefore, it divides the order into two components. One component connects the London endpoint to an unspecified NNI. The other component connects the Roman endpoint to another unspecified NNI.*
>
> *The primary SP assigns one service component to a secondary SP in England and the other to a secondary SP in Italy. It updates the Italian service component, specifying it must provide an NNI that interfaces with the English SP's network.*
>
> *Using the OSLP subcontracting service, the primary SP creates a new order representing each service component and binds the new orders to the original order from the MSP. The primary SP also "delays" transmission of the English SP's order.*
>
> *The OSLP subcontracting layer records the original order and the new orders in its local data base. It sends one new order to the secondary SP in Italy and defers transmission of the other new order.*
>
> *As the secondary SP in Italy provisions its component, it sends status updates messages to the primary SP. When the primary SP's subcontracting layer receives these messages, it re-evaluates the status of the original order from the MSP.*
>
> *The primary SP's subcontracting layer composes a status update message that includes information concerning the original order, as well as the two new orders. The primary SP's subcontracting layer sends this message to the MSP as well as to the primary SP's higher lever applications.*
>
> *When the primary SP application detects that the secondary SP in Italy has assigned an NNI, it updates the order to the secondary SP in England and removes the delay. The primary SP's subcontracting layer sends the order to the SP in England and order processing continues.*

OSLP Architecture

The following are OSLP architectural goals:

- Maximize the number of SPs that can partake in the protocol
- Maximize the number of views through which SPs can access each other's ordering information using the protocol.

These goals are in conflict. As the protocol becomes more robust, and offers a larger number of access views, it also becomes more difficult and costly to integrate with SP legacy systems. As the protocol becomes more difficult and costly to integrate with SP legacy systems, the number of SPs that can partake in the protocol decreases.

Therefore, OSLP provides the minimum number of access views required to support order management. Specifically, OSLP implements a multi-state protocol.

Multi-state protocols support the following scenario:

- The MSP creates an order within its own data base of record.
- The MSP sends a message to the SP informing the SP that the new order has been created.
- The SP creates an identical order within its data base of record.
- The SP processes the order contained by its data base of record according to local policy. It updates order status as appropriate.
- Each time the SP updates the order contained by its data base of record, it sends a status update message to the MSP. The status update message contains all SP provided order attributes, including status.
- The MSP receives the status update message and updates the corresponding order in its data base of record.
- When the MSP receives a status update message that indicates the order has been completed, the MSP sends the SP a message indicating whether or not it is satisfied with the service rendered.

Once the MSP sends its initial message to the SP, the MSP's ability to communicate with the order contained by the SP data base of record is extremely limited. Specifically, the MSP can:

- Monitor status update messages sent from the SP
- Synchronize its copy of the order with the SP copy by requesting a retransmission of the last status update message.
- Send a message to the SP that appends free form text to the order. The message can also escalate an ordering issue.

In order to modify or cancel an order, the MSP must create a new order within its data base of record. The new order supersedes or cancels the original order. The MSP sends a message to the SP and the SP creates an identical order within its data base of record. The SP processes both orders according to the scenario described above.

Multi-state protocols minimized the cost of integration with legacy systems. SPs execute only a few course-grained operations upon orders that reside in each others SP's domain. Therefore, SPs need not implement complex adaptation functions that map OSLP messages to every possible view of their legacy data base of record. SPs implement only a small set of adaptation functions that support order creation, status update reporting, and order closure.

The OSLP Protocol Data Unit (PDU)

Organizations initiate and progress orders by exchanging OSLP PDUs. In OSLP, the initiating organization assumes the *MSP role* and the order recipient assumes the *SP role*.[2]

[2] *The distinction between the MSP organization and the MSP role is important. Recall that in a subcontracting arrangement, the primary SP assumes the SP role with respect to the original contract and the MSP role with respect to the subcontract (see Figure 2)*

The OSLP PDU contains a header and a payload. The header represents service independent order attributes while the payload represents a list of specific services that are being ordered.

Most header values are supplied by the MSP role at order creation time. Selected attributes are supplied by the SP role. Table 1 describes the OSLP header fields.

The following are OSLP message types:

- OSLP Service Request (OSR)
- OSLP Provisioning Status Update (OPSU)
- OSLP Ping (OPNG)
- OSLP Provisioning Completion Confirmation (OPCC)
- OSLP Re-engineering Notification (OREN)
- OSLP Free Form Text (OTXT).

The OSR Message

The MSP role creates a service order within the SP data base of record by sending an OSR messages.

The OPSU Message

The SP role sends the MSP role one or more OSPU messages between OSR reception and order completion. The SP role sends an unsolicited OSPU message each time it achieves a provisioning milestone. It also sends an OPSU message in response to each status inquiry received from the MSP role.

The OPSU message specifies order status and engineering details. Table 2 defines order status values.

The OSLP Ping Message (OPNG)

The MSP role sends the SP role an OPNG message order to request a status update regarding a particular order. The SP responds to the OPNG message with an OPSU message.

OPSU Provisioning Completion Confirmation (OPCC) Message

The MSP role sends the SP role an OPCC message in response to an OPSU (complete) message. The OPCC message indicates whether or not the MSP role concurs that the service has been provisioned as requested. Upon receiving a

Table 1 OSLP Header Attributes

Header field description	Supplied by:
Organization identifier for the MSP role	MSP role
Order identifier assigned by the MSP role	MSP role
List of contacts that represent the MSP role	MSP role
Organization identifier for the SP role	MSP role
Order identifier assigned by the SP role	SP role
List of contacts that represent the SP role	SP role
Organization identifier for the MSP	MSP role
Order identifier generated by MSP	MSP role
Order type (i.e., request bid with pricing information, request bid without pricing information or request provisioning)	MSP role
Order disposition (i.e., new order, supersede previous order, cancel previous order)	MSP role
OSLP message type (See details below)	MSP role
Order due date	MSP role
Order estimated completion date	SP role
Order status (See details below)	SP role
Date and time of last status update	SP role
Order confirmation status (See details below)	MSP role
Subcontracting details (i.e., a list of orders sent to secondary SPs in order to fulfil current order)	SP role
List of annotations to current order	MSP or SP role

Table 2 OSPU Service Request Status Values

Status	Description
Initial	The order has been created, but not acted upon.
Claimed	The SP role has determined that it can provide the entire locally, without subcontracting.
Distributed	The SP role has determined that it cannot provide the entire service without subcontracting. Therefore, the SP role has subcontracted at least a portion of the request to one or more secondary SPs.
Subcontracted	The SP role has determined that it cannot provide the entire service without subcontracting. Therefore, the SP role has subcontracted at least a portion of the request to one or more secondary SPs. The secondary SPs have claimed their portion of the order.
Engineered	The SP role has engineered the service request (e.g., assigned access ports).
Complete	The SP role has provisioned and tested the service.
Accepted	The MSP role has accepted service delivery. Billing may begin.
Failed	The SP role has determined that it cannot provide the service.
Aborted	The SP role has aborted the service request at the MSP’s request.
Unknown	The SP role has received a status inquiry for an unknown order.

positive OPCC message, the SP role initiates the billing process. Upon receiving a negative OPCC message, the MSP and SP roles resolve the issue outside of OSLP.

If the SP role does not receive an OPCC message (either positive or negative) within a configurable period, it assumes that the service has been provisioned as requested and initiates the billing process.

The OSLP Re-engineering (OREN) Message

The SP role sends the MSP role an OREN message to indicate a unilateral change in engineering details after transmission of the OPSU (complete) message. Changes can represent network grooming efforts or emergency repair actions.

OREN messages are propagated upstream to the MSP organization.

The OSLP Free Text Message (OTXT)

The OTXT message can serve any of the following functions:

- Carry free form text regarding an order
- Manage escalation of an issue to SP role management.

Either the MSP or SP role can send an OTXT message at any time.

OSLP Stack

OSLP is a multi-state protocol that does not require robust interactive communications services. Therefore, OSLP does not rely upon services provided by the Common Information Service Element (CMISE).

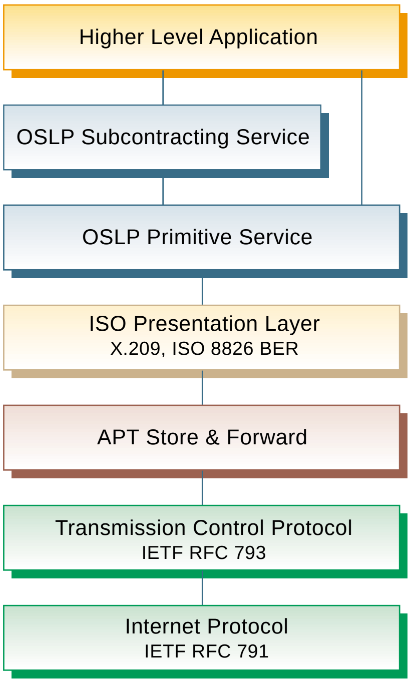


Figure 5 OSLP Supporting Stack

Furthermore, OSLP does not require object location services. Therefore, OSLP also does not rely upon services provided by X.500 or the Common Object Request Broker Architecture (CORBA).

Alternatively, OSLP relies upon services provided by the stack depicted in Figure 5.

The Basic Encoding Rules (BER) described in ITU-T Recommendation X.209 provide OSI presentation layer services.

Point-to-point store and forward services are provided by the Alliance Point-to-Point Transport Protocol (APT). APT is defined in the OSLP documentation suite.

Lower layer services are provided by the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

As stated in Recommendation M.3320 [3], SPs maintain pair-wise agreements regarding encryption and security administration. Because each SP is constrained by its national security and encryption policy, it may not be possible to specify a single, global security policy.

Conclusions

OSLP provides a mechanism through which SPs exchange ordering information. As OSLP's architecture minimizes the cost of integration with legacy order management systems, it encourages a wide community of SPs to partake in the protocol.

References

1 Bonica, R P. Alliance Approach to the TMN X Interface. *Telektronikk,* 94 (1), 39–47, 1998 (this issue).

2 ITU-T. *Principles for a Telecommunications Management Network.* Geneva, ITU, 1992. (ITU-T Recommendation M.3010.)

3 ITU-T. *Management Requirements Framework for the TMN X Interface.* Geneva, ITU, 1997. (ITU-T Recommendation M.3320.)

Ronald P. Bonica is a member of the MCI Internet Engineering Team. He is currently under contract to the Concert Frame Relay Team and participating in the Concert Alliance Interface Standards effort. His interests are TMN and Internet architecture.

e-mail:
rbonica@mci.net

Introduction to RM-ODP
– A Reference Model of Open Distributed Processing

HÅKON LØNSETHAGEN

Distributed processing has become increasingly important in the last decade, due to the need for interconnecting existing systems, new organisational trends, and intra- as well as inter-organisational co-operation. RM-ODP aims at providing a co-ordinating framework for the standardisation of Open Distributed Processing.

1 Introduction

The joint ISO/ITU standardisation of RM-ODP, Reference Model of Open Distributed Processing, aims at providing a co-ordinating framework for the standardisation of Open Distributed Processing. This framework identifies an architecture that supports distribution, interworking and portability. These factors are the basis for providing the benefits of distributed information processing in an environment of heterogeneous enabling technology. The main users of RM-ODP are considered to be standards writers and architects of open distributed systems.

Besides distribution of possibly remote processes, distributed systems are characterised by the following inherent characteristics [1]:

- *Concurrency*, i.e. components executing in parallel
- *Lack of global state*, i.e. the entire global state of a distributed system cannot be precisely determined
- *Partial failures*, i.e. any component may fail independently of any other components
- *Asynchronisity*, i.e. communication and processing activities are not driven by a single global clock, and related events in a distributed system cannot be assumed to take place at a single instant.

With these aspects taken into account, RM-ODP must in addition to heterogeneity, also consider and provide for autonomy, openness (enabling both portability and interworking), integration (incorporating various systems and resources into a whole), and flexibility. The issue of autonomy deals with the situation where the systems are spread over a number of autonomous management or control authorities (organisational units or administrative domains), without any central point of control. Flexibility, addresses several issues, such as continued operation of legacy systems, support of a system's evolution, as well as capabilities facilitating run-time changes and reconfigurations.

The RM-ODP standards are organised into four parts.

- Part 1 [1], *Overview;* provides an overview of RM-ODP, including its motivation (partly covered above), scope, justification and explanation of key concepts, and an outline of the architecture, including explanatory material. This part is not normative.
- Part 2 [2], *Foundations;* contains the definition of concepts and analytical framework for normalised description of (arbitrary) distributed processing systems. The level of detail is limited to what is necessary and sufficient as a basis for Part 3. This part is normative.
- Part 3 [3], *Architecture;* contains the specification of the required characteristics that qualify distributed processing as open. These are constraints to which ODP standards must conform. It uses the descriptive techniques from Part 2. This part is normative.
- Part 4 [4], *Architectural semantics;* contains a formalisation of the central ODP concepts defined in Part 2. The formalisation is achieved by interpreting each concept in terms of constructs of different standardised formal description techniques. This part is normative.

This paper, whose aim is to provide an introduction to RM-ODP, is predominantly based on these four parts. In addition, this article also attempts to identify and discuss some of the challenges and still open issues related to RM-ODP.

The development of RM-ODP has been based on several sources [5], of which the ANSA Research Program has made a significant contribution. The idea of dividing the problem domain of open distributed processing into five projections to handle complexity, is discussed in [6]. These projections, which correspond to the five viewpoints of RM-ODP, is a key element in the ANSA Architecture, consisting of a set of six frameworks addressing most aspects of distributed systems design and implementation.

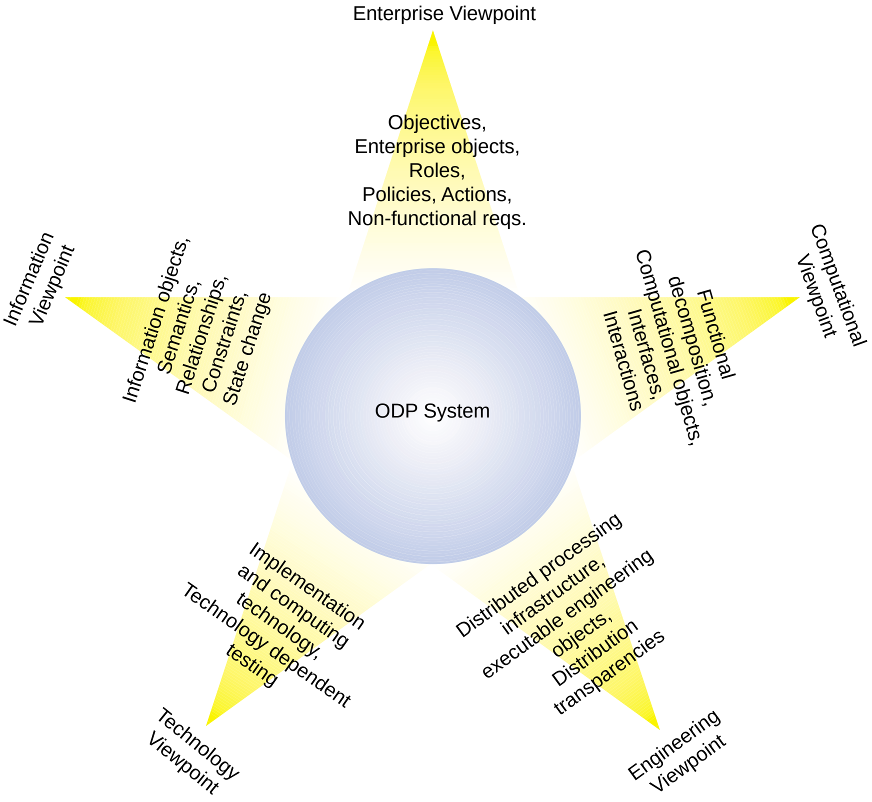

Enterprise Viewpoint
Objectives, Enterprise objects, Roles, Policies, Actions, Non-functional reqs.
Information Viewpoint
Information objects, Semantics, Relationships, Constraints, State change
Computational Viewpoint
Functional decomposition, Computational objects, Interfaces, Interactions
ODP System
Implementation and computing technology, Technology dependent testing
Technology Viewpoint
Distributed processing infrastructure, executable engineering objects, Distribution transparencies
Engineering Viewpoint

Figure 1 RM-ODP Viewpoints

RM-ODP Part 1 states: "A viewpoint is a subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the design of the system". The five RM-ODP viewpoints (viewpoints on the system and its environment) are (se also Figure 1):

- the *enterprise viewpoint*, which is concerned with the business activities of the specified system, by focusing on purpose, scope, and policies
- the *information viewpoint*, which is concerned with the information that needs to be stored and processed, and the semantics of the information
- the *computational viewpoint*, which is concerned with distribution by addressing functional decomposition into computational objects which interact at interfaces
- the *engineering viewpoint*, which is concerned with the mechanisms and functions supporting distribution of the computational objects and their interaction
- the *technology viewpoint*, which is concerned with the choice of implementation and computing technology.

The viewpoints should not be considered as layers of functionality, nor should a fixed order be assigned to them according to design methodology. In addition to the five viewpoints, the framework defined by RM-ODP comprises a viewpoint language for each viewpoint, specification of functions required to support ODP systems, and a set of transparency prescriptions. The latter shows how to use the ODP functions to achieve distribution transparency. Each viewpoint language defines the concepts and rules for specifying ODP systems from the corresponding viewpoint. The combination of the computational language, the engineering language, and the transparency prescriptions determine the architecture of the ODP systems.

A goal of the framework is to allow different parts of the design to be worked on separately. However, it should also clearly identify those parts of the design that constrain other parts. The mechanism of viewpoints is one of the two main structuring approaches adopted by the ODP architecture, while distribution transparencies provide the other.

Distribution transparencies deal with a number of concerns and aspects that are direct results of distribution, such as remoteness, failure, etc. The ODP framework addresses these concerns by identifying generic means to make these aspects transparent to the application designers and developers. RM-ODP defines the following transparencies (along with related requirements and solutions that satisfy them):

- *access transparency*, – masks the differences in invocation mechanisms and data (message) representation to enable interworking between distributed processes of heterogeneous technology. This transparency is generally inherent to the distributed infrastructure.
- *failure transparency*, – masks from an object (see below for a definition of 'object' and 'interface') the failure and possible recovery of other objects (or itself) to enable fault tolerance. If this transparency is provided, the designer can work under the assumption that this class of failure does not occur.
- *location transparency*, – masks the use of information about location in space when identifying and making association between object interfaces. When this transparency is provided, identification of object interfaces is achieved by naming based only on a logical view.
- *migration transparency*, – masks from an object the ability of a system to change the location of that object. Migration of objects is often used to achieve load balancing and reduce latency.
- *relocation transparency*, – masks from an interface the relocation of another interface bound to it. Relocation allows system operation to continue even when migration or replacement of some objects creates transient inconsistency.
- *replication transparency*, – masks the use of a group of mutually behaviourally compatible and synchronised objects to support an interface. Replication is often used to enhance performance and availability.
- *persistence transparency*, – masks from an object the deactivation and reactivation of other objects (or itself). The state of an object must be persistently stored even when processing and memory resources cannot be allocated to the object, or as a means to facilitate recovery.
- *transaction transparency*, – masks from an object the co-ordination of activities amongst a configuration of dependent objects to achieve consistency.

The transparencies may be provided for by a generic distributed infrastructure. This infrastructure or distributed processing environment, provides ODP functions coping with the transparencies. By implementing these functions as part of a generic infrastructure, software reuse is achieved by letting the ODP functions be (re-)used by several applications. The notion of middleware, which in the last few years has become an established term in the industry, is used to designate software providing the generic distributed infrastructure. Various middlewares, however, may offer a subset of the above list of transparencies, as well as additional functions and services.

The remainder of this paper is structured as follows: Section 2 briefly presents the fundamental and generic (modelling) concepts of RM-ODP. Sections 3, 4, 5, 6, and 8 present the five RM-ODP viewpoints and corresponding viewpoint languages, in the order as indicated above. In addition, distribution transparencies and ODP functions are covered in more detail in section 7. Section 9 provides an introduction to the issue of architectural semantics. A discussion is provided in section 10, on what is believed to be open issues related to RM-ODP. Concluding remarks are given in section 11. An example of a way of using RM-ODP is provided in a separate paper in this issue of *Telektronikk* [7].

2 Foundations – The Common Modelling Concepts

The ODP Architecture[1] is defined in Part 3 of RM-ODP. It defies a viewpoint language for each of the five viewpoints. The concepts identified in each of the viewpoint languages are based on and possibly refine the generic modelling concepts defined in Part 2, Foundations.

In addition to providing interpretation of a few terms or concepts generally applicable to any modelling activity, Part 2 makes the following categorisation of concepts;

- *basic modelling concepts*, – concepts for modelling the ODP systems. Any viewpoint language must relate to these concepts.
- *specification concepts*, – concepts related to the relevant specification language. These concepts are not intrinsic to the distributed system, however, some systems may directly rely on specifications of for example a conceptual specification.
- *structuring concepts*, – concepts that relate to various aspects of designing and description of distributed systems, such as their various system architecture properties and policies, organisation of objects, naming issues, issues related to activity and behaviour, as well as management.
- *conformance concept*, – concepts necessary to interpret the notion of conformance to ODP based standards and of conformance testing.

2.1 Basic modelling concepts

By the basic modelling concepts, ODP introduces a general object-based model. However, this does not imply that the object-oriented modelling style is adopted[2]. An object is an abstract model of an entity. An object is further characterised by its unique identity, by its state, its encapsulation, and its behaviour. Encapsulation ensures that the state of the object may only be accessed via one or more interfaces of an object. Otherwise, only internal actions may alter the state of an object.

At this generic level, few assumptions are made, and an object may be of any level of granularity, complexity, and allow internal parallelism. Interactions between objects are not constrained and may include both asynchronous (which implies concurrency) and various synchronous interactions.

Interactions among objects may only occur on the object interfaces. An interface identifies a set of possible interaction types, and may be considered as a service of the object. This set of possible interactions represents a part of the object’s behaviour. An object may have multiple interfaces, which allow functional separation as well as distribution of the object’s points of interaction.

The above mentioned properties of an object, provide well defined separation between objects. That is, every dependency among objects is captured by a corresponding interface. This further allows a compatible object, as perceived by its environment, to replace another object without influencing the environment of the original object.

2.2 Specification Concepts

While an ODP system in general can be described by the basic modelling concepts as a collection of related, interacting objects, the specification concepts introduce several additional concepts that may be related to requirements placed on any specification language used for the specification of an ODP system.

Among these concepts are the important notions of composition and refinement. Composition and its dual, decomposition, of specifications can be a useful means of the modelling and design process. A distinction between composition of objects and composition of behaviours is made. Refinement is the process of transforming one specification into another more detailed specification.

Furthermore, the notions of type and class are defined, as well as the corresponding notions admitting hierarchies of types or classes, i.e. subtype/supertype and subclass/superclass. The notions of template and instantiation of a template also fall into the category of specification concepts.

2.3 Structuring Concepts

While the specification concepts allow structuring of specifications, the structuring concepts on the other hand are concerned with structuring and properties of the ODP systems as such.

In addition to various concepts used for organising and relating sets of objects, such as configuration, domain, group, reference point, and epoch, this part also defines several fundamental concepts related to naming, and transparencies, as well as policies – the latter will be covered in more detail with respect to the enterprise viewpoint. Management concepts such as application management, communication management, management information, managed and managing roles, and notification are also defined.

An important policy concept is that of environment contract, which is a contract between an object and its environment. Constraints related to quality of service (QoS), usage, and management are introduced. These notions are important to the enterprise and the computational viewpoints. QoS constraints include temporal, volume (e.g. throughput), and dependability constraints (e.g. availability, reliability, maintainability, security and safety). Usage and management constraints include issues related to location as well as distribution transparencies. Non-functional requirements is a frequently used expression denoting these constraints.

While the concepts of behaviour and activity were identified as basic modelling concepts, concepts related to structures of activities, as well as concepts related to contractual behaviour (related to contracts among sets of objects) are defined in this part. Several categories of roles are also identified related to causality of object interaction. Causality, e.g. producer vs. consumer, is an issue particularly relevant to the computational viewpoint.

Of particular importance in open distributed processing is the concept of binding between object interfaces. A binding is a contractual context, which is the knowledge that a particular contract is established between two or more interfaces, and hence between their supporting objects, and implies that a particular behaviour of this set of objects is expected.

[1] *A general description of the notion of architecture at this level of abstraction was not found in RM-ODP Part 2. However, it becomes evident from the usage of terms in the documents that while the ‘ODP (Architectural) Framework’ identifies the (frame of) overall elements encompassed by the ODP Architecture, the notion of ODP Architecture itself designates both the framework and the concepts and rules defined as part of each element of the framework.*

[2] *An object-oriented modelling style must include the concept of inheritance [8] and possibly polymorphism (generic operation) [9].*

2.4 Conformance Concepts

Distributed systems are potentially very complex, the components are heterogeneous and may undergo individual evolution, and several administrative domains may be involved. This complexity puts a great challenge on expression of conformance requirements and on conformance assessment and testing.

Conformance relates statements of a specification (a standard) to observable events and behaviour of the corresponding implementation at certain conformance points. The RM-ODP identifies several classes of reference points in the architecture. Such a reference point is a potential conformance point. If so declared by some specification, this point must then be accessible to testing of conformance.

RM-ODP identifies four classes of reference points of which a reference point potentially may be identified as a conformance point:

- *programmatic reference point*, – at which a programmatic interface can be established to allow access to a function. A programmatic interface is an interface which is realised through a programming language binding.
- *perceptual reference point*, – at which there is some interaction between the system and the physical world, e.g. a human-computer interface.
- *interworking reference point*, – at which an interface can be established to allow communication between two or more systems or components. Interworking conformance involves interconnection of reference points.
- *interchange reference points*, – at which an external physical storage medium can be introduced into the system. The storage medium may be used to interchange information from one system to another.

3 The Enterprise Viewpoint and Language

The enterprise viewpoint allows an ODP system to be represented in the context of the enterprise in which it operates. The aim is to identify requirements of a system in terms of objectives, policy statements and obligations by the various roles identified in the enterprise viewpoint. This viewpoint further identifies activities to be realised by the system as well as the various contracts with its environment. The motivation for a separate viewpoint to express these issues, is based on the idea of decoupling the set of objectives for the system from the way they are realised.

The concepts of ‘community’ and ‘federation’ are defined by this viewpoint language. A community represents the ODP system and the environment in which it operates. A community will identify the scope of the system relative to this community. The following elements constitute a community:

- the enterprise objects comprising the community
- the roles fulfilled by each of those objects
- policies governing interactions between enterprise objects
- policies governing the creation, deletion, and usage of resources by enterprise objects
- policies governing the configuration and structuring of enterprise objects and assignment of roles to enterprise objects
- policies relating to environment contracts, governing the system.

An enterprise object can fulfil roles associated with different communities. This is determined by the environment contracts on which a corresponding community is based. An enterprise object fulfilling a role is defined and constrained by permissions, obligations, and prohibitions associated with that role. These aspects, expressing policies of an object may change dynamically according to interactions between objects. In the enterprise viewpoint, actions between objects relating (and changing) obligations, permissions, and prohibitions to associations between objects are of particular interest. Furthermore, objectives and policies related to resource usage, accounting, and non-functional requirements are identified within the enterprise viewpoint.

Enterprise objects may be owned and controlled by different authorities or administrative domains. A federation is a particular community where objects are associated with different authorities. The objective of a federation will typically depend on joint and co-ordinated activities. The autonomy of enterprise objects must be considered when describing the rules for the establishment of a federation, the joining to a federation, as well as the cessation of federation participation. The policies related to delegation, security and trust are of particular interest to communities describing federations.

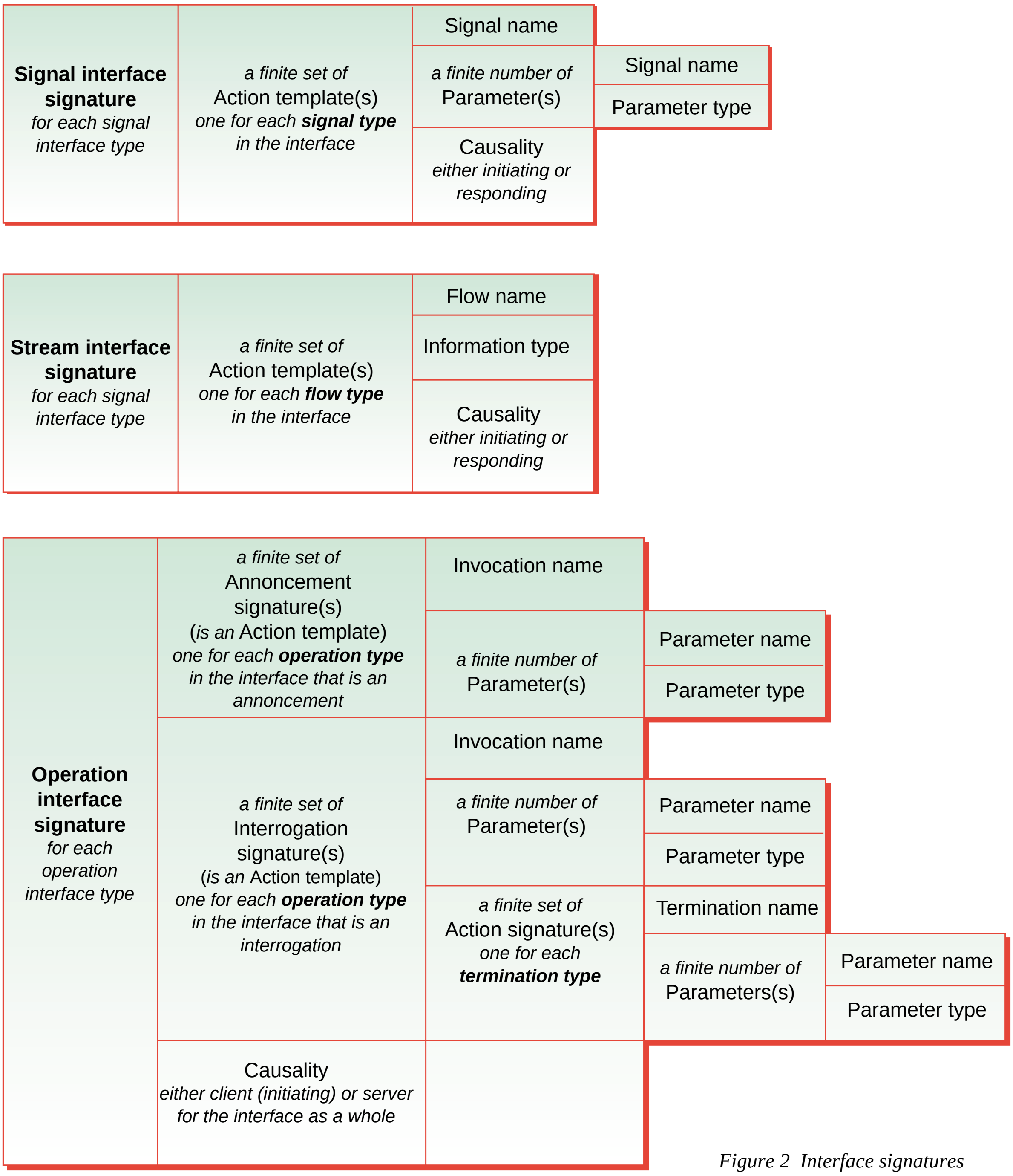
All four classes of reference points, as described in section 2.4, may be subject to enterprise specification. Conformance statements in the enterprise language in terms of a particular set of objectives and policies must be related to a reference point identifiable in the engineering viewpoint. By observing the behaviour of the system at this reference point, the interactions with the system may be interpreted in the enterprise language terms to check conformance.

4 The Information Viewpoint and Language

The information viewpoint is focused toward the information that needs to be stored and processed by the system, that is, the semantics of this information as well as the semantics of the information processing in an ODP system. The information language defines the generic concepts, rules and structures to be used in this viewpoint. An information specification defines the information and semantics in terms of a configuration of information objects and their relationships, the behaviour of those objects, and the environment contracts for the system.

Furthermore, the motivation for the information language is to facilitate the correct interpretation of information conveyed while interacting among components of an ODP system. If components of a distributed system are not founded on a common understanding of the common information among the communicating components, the system is not likely to behave as expected.

The information objects represent entities that occur in the real world, in the ODP system, or in other viewpoints. An *atomic* information object represents a basic information element, while complex information is represented as a composite information object containing encapsulated information objects. Hence, an information object being a component of a composite object cannot be a component of another.



Signal interface signature *for each signal interface type*

a finite set of Action template(s) *one for each* ***signal type*** *in the interface*

Signal name

a finite number of Parameter(s)

Signal name

Parameter type

Causality *either initiating or responding*

Stream interface signature *for each signal interface type*

a finite set of Action template(s) *one for each* ***flow type*** *in the interface*

Flow name

Information type

Causality *either initiating or responding*

Operation interface signature *for each operation interface type*

a finite set of Annoncement signature(s) (*is an* Action template) *one for each* ***operation type*** *in the interface that is an annoncement*

Invocation name

a finite number of Parameter(s)

Parameter name

Parameter type

a finite set of Interrogation signature(s) (*is an* Action template) *one for each* ***operation type*** *in the interface that is an interrogation*

Invocation name

a finite number of Parameter(s)

Parameter name

Parameter type

a finite set of Action signature(s) *one for each* ***termination type***

Termination name

a finite number of Parameters(s)

Parameter name

Parameter type

Causality *either client (initiating) or server for the interface as a whole*

Figure 2 Interface signatures

The information language comprises three schemata. An information object template may reference any of these schemata:

- *invariant schema*, – a set of predicates on one or more information objects which must always be true for the entire life-time of the objects. Thus, the invariant schema constrains the possible states and state changes of the corresponding objects.
- *static schema*, – the state of one or more information objects, at some point in time, subject to the constraints of any invariant schema.
- *dynamic schema*, – the allowable state changes of one or more information objects, subject to the constraints of any invariant schema.

Typically, the identified state change will go from one static schema to another static schema, and as such, model the behaviour of the system in terms of the state space of which it operates. Allowable state changes can be subject to ordering and further temporal constraints. A dynamic schema can involve the creation of new information objects or the deletion of information objects.

While the invariant schema may be regarded as the specification of the types that will always be satisfied of one or more information objects, the static schema is the specification of the types of one or more information objects at some particular point in time. The types specified by the static schema are thus subtypes of the types specified in the invariant schema.

Distribution and localisation are not considered by the information specification. However, the notion of interface is not prohibited as part of an information specification. No assumptions may, how-

Box 1 – Binding between computational interfaces

A binding between computational interfaces is an abstraction of an established communication path between the interfaces. The notion of *explicit binding* is used when explicit actions are needed to establish a binding. An explicit *binding action* allows explicit handling of the binding, including means to express configuration and QoS. *Implicit bindings* on the other hand, are assumed when the specification notation does not offer means of expressing binding actions. Implicit binding is only defined for operation interfaces. It involves creation of the appropriate client operation interface, binding the client interface to the server interface, and invocation of the server object, using the client operation interface.

Any binding action must ensure that the bound interfaces are of the same kind, e.g. operational, and that they have complementary causality, e.g. client interface against a server interface. Moreover, the types of the signatures must also be compatible.

Explicit bindings are either primitive bindings or compound bindings. A *primitive binding* is a binding directly between two computational objects. The corresponding binding action is parameterised using one identifier of each of the two interfaces involved. One of these interfaces is an interface belonging to the computational object initiating the binding. A primitive binding action is atomic in the sense that either the binding will be established or it will fail.

A compound binding involves a dedicated binding object. A binding object is itself a computational object, however, constrained by a set of rules for compound bindings. A binding object is a means to bind more than two computational objects, and further to allow controlling the configuration of the binding as well as controlling QoS related aspects.

By appropriate parameters, the compound binding action identifies the desired binding object template to be used and the interfaces of the computational objects to take part in the compound binding. The interfaces to be bound must be assigned roles corresponding to the formal role parameters of the binding object template. The behaviour of the binding object is expressed in terms of these roles.

The initiator of the compound binding action may in subsequent interactions with the other computational objects disseminate the identifier of the binding object control interface and information about the state of the binding object. At that time, the other computational objects can also access the control interfaces of the binding object and manipulate the binding object.

ever, be made about the location of the interface or about the interface appearing as such in an implementation.

Conformance relative to a conformance point must be based on observations at some engineering reference point(s). The behaviour at this reference point must be consistent with a particular set of invariant, static, and dynamic schemata as identified by the corresponding conformance statements.

5 The Computational Viewpoint and Language

The computational language and the engineering language have got the most attention in the RM-ODP specifications. While the enterprise and information viewpoints deal with distribution at a high level only, that is, what constitutes a system, the computational as well as the engineering viewpoints explicitly addresses distribution. The computational viewpoint is concerned with the decisions of how to distribute the job of the system by a functional decomposition into computational objects. On the other hand, the mechanisms for interaction among distributed components are provided by the engineering viewpoint.

The computational language identifies an object model, which in broad terms defines what kinds of interfaces an object can have, how interfaces may be bound to each other, and the kinds of interactions that may take place at the interfaces. The goal of the computational language is to enable a component specifier to express constraints on distribution of an application only in terms of environment contracts, i.e. interfaces and interface bindings. The degree of distribution, e.g. replication, or any other dependency on realisation of distribution need not be expressed in the computational language. This will promote open interworking and portability of components.

5.1 Computational interfaces and object interactions

The computational language identifies three types of interfaces; operational, stream, and signal interfaces. These categories of interfaces correspond with the three forms of interactions that can take place between computational objects. Each interaction must occur at one of the established (bond) interfaces of a computational object. The forms of interactions are:

- *signals,* which are the elementary atomic interactions. A signal is considered as an atomic action shared between an initiating object and a responding object, resulting in a one-way communication from the initiating object to the object accepting the communication.
- *operations,* which from a client object, request an invocation of some function at the server object. Furthermore, there are two types of operations;

  i) *interrogation,* where the server returns a response (termination) to the client object, and

  ii)*announcement,* where no response by the server back to the client is expected. An announcement may be used to report some event or state change.
- *flows,* which is an abstraction of a continuous sequence of interactions, resulting in conveyance of information from a producer object to a consumer object. Flows can be used to model, for example, audio and video streams as part of a multimedia telecommunications service.

Notice that the notion of client and server as associated with interrogations and announcements are merely roles. Thus, a server object as seen relative to an interrogation may emit an announcement to the corresponding client object. In that case, this client object takes the server role with respect to the mentioned announcement.

Whereas participants of a flow or an operation may have an inconsistent view of an interaction at different times, especially when failures have occurred, there is no concept of partial failure of signals. A signal either fails or succeeds identically for both participants in the interaction. Modelling operations and flows in terms of signals may be necessary in order to enable consideration of end-to-end quality of service (QoS) characteristics.

A computational specification will specify a set of computational object templates, which comprises a set of computational interface templates. Both the interface and the object templates comprise associated behaviour specifications and environment contract specifications. In addition, an interface template contains one of either a signal, an operational, or a stream interface signature. These signatures and their comprised elements can be considered the core of the computational language. Each of the interface signatures and their comprised elements are illustrated in Figure 2. The relationship between table cells is a 'comprises' relationship. The plural (s) indicate a 'many' cardinality of the relationship.

The notion of complementary interface, say Y is complementary to X, indicates that interface signatures are identical except for the complementary causality.

Parameters identifiable in a signal or operation signature can take values identifying computational interfaces or computational interface signature types. The latter then makes the type system of the computational signatures higher order.

A computational specification defines an initial configuration of computational objects and their behaviour. This configuration will change as computational objects or interfaces are instantiated or deleted, binding actions are performed, or control functions on binding objects are effected.

The computational language is associated with several categories of structuring rules. Various naming rules constrain the context and scope of names in computational specifications. Furthermore, the template rules identify the various kinds of actions performed by computational objects, the failure rules identifying potential points of failure in computational activities, and the portability rules identifying aspects that must be considered when designing a portability standard.

The structuring rules that relate to distribution transparent interworking are of particular importance. They are the interaction rules, which have been considered in this subsection, the binding rules, and the type rules. The latter two are described in the accompanying text boxes.

Box 2 – Interface signature typing and subtyping

Computational interfaces are strongly typed. This enables early (e.g. compile-time) consistency checking between interface specifications and implementations. The type rules in RM-ODP are focused on subtyping rules for the various kinds of computational interface signatures, i.e. signal, operation, and stream signatures.

The notion of subtyping is particularly important with respect to evolution of systems. A desired property is to enable replacement of an interface or a component (e.g. a computational object) without any noticeable difference to its environment. Usually, one would like to replace a server object without having to change (several) client objects. Replacement without any noticeable difference to the environment is the ideal goal related to the notion of substitutability. In Part 2, Foundations, this property is called behavioural compatibility. The notion of interface type and interface subtype ideally captures all aspects of an interface specification necessary to ensure substitutability; an interface may replace another interface if the first is an interface subtype related to the interface type of the second.

However, typing and subtyping in RM-ODP, as well as in other object models like the CORBA object model [10], are based on types associated with interface signatures only. The notion of interface signature types captures the syntactic aspects of the interface specifications as well as type consistency of the parameters of the interfaces. Additional aspects related to interface behaviour or environment contracts of the interface are not covered by the typing of interface signatures. Hence, the signature subtyping rules only define a set of minimum requirements for achieving substitutability.

The semantics of flows and composition of flows related to stream interfaces may be application dependent, thus RM-ODP does not prescribe a complete set of subtyping rules for stream interface signatures.

Intuitively one can say that the subtyping rules are based on the idea of avoiding unexpected kinds of interactions. As an example, considering operation interface signature types having the server causality[3]: *X* is a signature subtype of *Y*, if for every interrogation in *Y* there is a corresponding one in *X* with the same name, the same number and name of the corresponding parameters; and each parameter type in *Y* is a subtype of the corresponding parameter type in *X*; and for every termination in *Y*, there is a corresponding one in *X* with the same name, the same number and name of the corresponding parameters; and for every parameter result type associated with the termination in *Y*, the corresponding result type in *X* is a subtype of the one in *Y*.

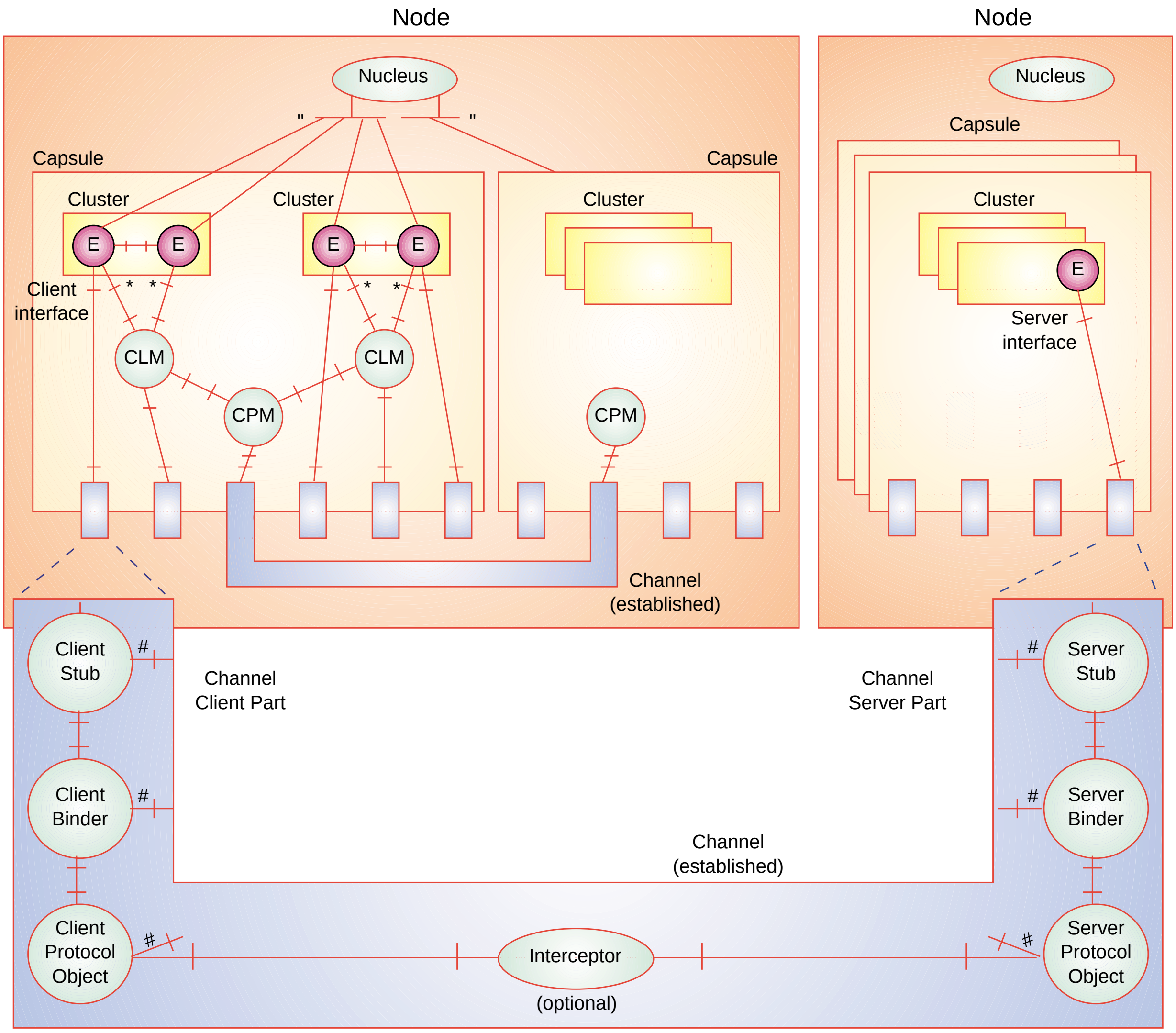
[3] *RM-ODP does not relate the subtyping rules for operation interfaces to the causality of the interface. The resulting problems this causes are pointed out by Sinnott and Turner in [11].*

6 The Engineering Viewpoint and Language

While the computational viewpoint is concerned with specification of abstract computational components or objects, the engineering language is concerned with how to realise these abstract objects and how to enable interactions among these realised engineering objects.

The engineering language is based on a mapping where each computational object is realised as one or several *basic engineering objects*. One basic engineering object, or the set of basic engineering objects, realising a computational object, can only correspond to that computational object. Accordingly, there is a one-to-one correspondence between an engineering interface and a computational interface (except when engineering objects are replicated).

The engineering language also defines concepts and rules that model the distributed infrastructure, which enables exe-



E	:	Basic Engineering Object
CPM	:	Capsule manager
CLM	:	Cluster manager
*	:	Object management interface
"	:	Node management interface
#	:	Channel control interface

Figure 3 Example structure of engineering objects

cution of the basic engineering objects as well as interactions among these objects. The distribution transparencies are provided by this enabling infrastructure. The allocation of basic engineering objects to computing and storage facilities (system resources) is a concern of the engineering viewpoint. This issue is relevant when considering for example system structures and how to balance processing load as well as means for increasing resilience. This issue is the topic of ‘Engineering communicating systems’, another article in this issue of *Telektronikk* [12].

In addition to identifying the concept of basic engineering object, the engineering language identifies plain engineering objects that are fundamental to and part of the distributed computing infrastructure. These engineering objects have a direct mapping to the available system resources and model the relevant aspects of these resources according to the inherent nested structure of system resources. Figure 3 provides an illustration of plain engineering objects and their nested structure, and how basic engineering objects fit into this structure.

The notion of *cluster* is used to represent a configuration of basic engineering objects forming a unit for the purpose of various object or cluster management tasks. The clusters reside at the inner level of the nested structure of engineering objects. One of these tasks is instantiation of the cluster, based on a cluster template representing a configuration of basic engineering objects, the initial state of the objects and the initial bindings associated with these objects.

The task of checkpointing is also associated with the concept of cluster. Checkpointing a cluster results in a cluster template reflecting the state and structure of the objects contained in the cluster at the time of the checkpointing action. Cluster checkpoints further provide a basis for tasks such as deactivation, reactivation, recovery and migration of clusters. The tasks just mentioned are provided by the cluster manager plain engineering object.

At the next grouping level, the concept of a *capsule* is used as a container for a group of clusters. Considered in relation to an operating system, the capsule can be thought of as a protected process with its own protected address space. The cluster on the other hand can be thought of as a segment of the virtual memory containing objects controlled by the capsule. The capsule is usually the smallest unit of independent failure. The capsule contains a capsule management engineering object that manages all the cluster managers residing in the capsule. In addition to interaction with the cluster managers initiating the cluster management tasks, the capsule manager may have to interact with other ODP functions to fulfil its management policy.

A capsule forms a single unit for the purpose of encapsulation of processing and storage, and can thus be considered as occupying a portion of the processing capacity and memory space controlled and managed by an operating system. At the outer level of grouping, the concept of a *node* is used to designate the total processing, storage and software resources controlled by an operating system. The internal structure of the node, e.g. the number of processors, is of no concern to the engineering viewpoint.

In terms of the engineering language, a node encompasses a *nucleus* representing (the kernel of) the operating system of the node. A nucleus is an engineering object controlling and managing the capsules of the corresponding node. The nucleus provides a set of node management interfaces, one to each of the associated capsules. It provides basic services to the capsules such as providing communication resources and generation of unique identifiers for identification of engineering object interfaces.

Interactions among objects in the same cluster are efficiently provided by programming language and run-time system specific means. On the other hand, interaction among objects in different clusters involves a configuration of engineering objects called a *channel.* A channel provides support mechanisms that make it possible to cope with inter-node communication, as well as capsule failure and relocation of cluster from one capsule to another, possibly of different nodes. Three kinds of engineering objects are involved in the establishment of a channel. The *stub* is concerned with the marshalling and representation of information carried between the basic engineering objects. A stub further relies on a *binder* that maintains the integrity and state of the association with the peer object. A *protocol* engineering object provides and manages the actual communication of messages between itself and its peer protocol object. The bindings between basic engineering object supported by channels are called distributed bindings. Group communication and multicast involve multi-endpoint channels.

When a basic engineering object interface is created, a corresponding engineering interface reference is created. Associated with this reference is information about the signature type of the interface, a communication address to be used when establishing a binding to this interface, and information related to expected non-functional capabilities of the channel used for the binding. A binding endpoint identifier is used by the basic engineering objects, in the naming context of a capsule, for identification and selection of appropriate binding endpoints.

If a channel crosses an administrative domain boundary, the channel will likely include one or more interceptor engineering objects. The role of the interceptor object may be to check and enforce various policies associated with the interactions, translation of interface references, as well as checking of signature types of the conveyed engineering interface references.

Associated with the concepts and structures just mentioned are additional rules that form an integral part of the engineering language. An engineering specification must conform to these rules. In addition, an engineering specification must identify the necessary ODP functions to support the transparencies assumed by the computational specification, and to support the functional distribution of an ODP system.

7 Distribution Transparencies and ODP Functions

As mentioned in the introduction, the distribution transparencies are fundamental for making abstract specifications of distributed systems possible. A transparency schema is associated with a computational specification and defines the constraints on the mapping from a computational specification to an associated engineering specification, – the latter specifies how to realise the transparencies. This mapping expands the computational specification with additional behaviour, and with ODP functions explicitly realising the functionality masked by the transparencies.

Although described in a separate section, the ODP functions are – based on their roles – naturally associated with the engineering viewpoint. ODP functions can be realised as objects or components, i.e. groups of objects. RM-ODP indicates dependencies among various functions. However, RM-ODP leaves the precise specification of these facilities to other specification efforts. In the specification effort by OMG[4], their CORBA Object Services can be considered as examples of ODP functions. See another article on CORBA in this issue of *Telektronikk* [13].

Access and location transparencies are assumed to be provided by every distributed processing infrastructure. These transparencies are provided by the fundamental facets of the infrastructure, that is, the channel and facilities for handling location independent interface references. Other transparencies can be provided in combination with these, however, based on selection by the application developer.

ODP functions can themselves be structured and complex, and may be modelled and specified based on the RM-ODP itself. Depending on the objective of the function, viewpoints such as enterprise and information may be more or less relevant. The computational viewpoint, however, is relevant for specifying ODP functions. In a computational specification of an ODP function, one can of course not assume the transparency to be provided by the function to be specified. When this function is relied upon as being part of the infrastructure, an abstraction of the function may be used in a computational specification. Some of the ODP functions identified do not have direct correspondence with a distribution transparency. However, these functions provide useful services that can naturally be associated with the infrastructure and reused by several applications.

[4] *http://www.omg.org/*

The ODP functions are placed into four categories:

- *Management functions*
  These are the node management, object management, cluster management and capsule management functions. These functions form an integral part of the engineering viewpoint.

Many of the further functions below are directly involved in supporting one or more transparencies. With one exception, these functions can be included in the infrastructure on a selective basis.

- *Co-ordination functions*
  The engineering interface reference function is an integral part of the engineering language. The other functions in this category are the event notification, the checkpointing and recovery, the deactivation and reactivation, the group, the replication, the migration, and the transaction function.
- *Repository function*
  The functions in this category are the storage, the information organisation, the relocation, the type repository, and the trading function. The trading function is supported by a trader object allowing other objects to announce their capabilities. Client objects can then search the trader for interface references according to requested services.
- *Security functions*
  They are the access control, the security audit, the authentication, the integrity, the confidentiality, the non-repudiation, and the key management function.

8 Technology Viewpoint

A technology specification defines the choice and suitability of technology for the actual implementation of the ODP system. It will include specification of basic components of enabling technology possible based on implementable standards as well as specification of the communication mechanisms and technology used. The choice of technology will influence the achievable quality of service and other aspects related to performance.

The issues relevant in the technology viewpoint can to a great extent be treated separately from the ODP systems development process as such. However, the technology language plays an important role when it comes to testing of ODP systems. The exact technology dependent conformance points are identified in this viewpoint, and associated with conformance points and specifications in the other viewpoints. Additional information must be supplied in this viewpoint enabling correct interpretation of observations in terms of the vocabulary of specifications from other viewpoints. The additional information required for testing is called IXIT – Implementation eXtra Information for Testing.

9 Architectural semantics

As the above text indicates, the RM-ODP architecture is a rich collection of concepts, their relationships, and rules aiming at covering every significant aspect of distributed processing. Ideally, the ultimate goal of this standardisation effort should be to reach precise and sound definitions of every architectural element, that is, to develop an architectural semantics. To achieve this, one must formalise and provide precise semantics for every architectural element, covering every viewpoint as well as relationships between viewpoints.

The architectural semantics work has been based on interpreting given architectural concepts in terms of the semantic model of a given formal description technique (FDT), or directly in (established) mathematical terms.

This formalisation process is expected to be beneficial in many respects. Sinnott and Turner provide an account of the benefits in addition to a discussion on various aspects related to the idea of architectural semantics [14]. The obvious benefit is the avoidance of any ambiguity associated with the architectural elements. This process also provides an opportunity to compare the chosen FDTs, and possibly suggest improvements to any of the FDTs. Furthermore, if architectural semantics for each viewpoint language is developed, this will provide a useful starting point for using the particular FDT to specify an ODP system from the considered viewpoint.

After reducing the ambitions as compared to the initial schedule [14], the architectural semantics work group has recently put forward Part 4, Architectural Semantics, for approval as an International Standard. In this version the architectural semantics cover the basic modelling concepts and the specification concept of Part 2, Foundations. Amendments to Part 4 can be expected, as developments for the viewpoint languages and a greater extent of the Part 2 concepts become more complete and mature.

10 Discussion

The work with RM-ODP has come to a milestone as Part 2 and Part 3 of RM-ODP have reached the level of international standard/recommendation in 1995. However, it is expected that new ODP standards will be developed to address specific areas of concern [15]. These efforts may in turn result in the need for revising the RM-ODP architecture, as well. The following will raise a few issues indicating where RM-ODP can be made more specific. However, as will be explained, this may depend on the intended use of the ODP architecture. It is not within the scope of this paper to go into detailed discussions of these issues – merely an introduction will be given.

10.1 Viewpoint correspondence and viewpoint relevance

The goal of the RM-ODP framework and architecture is to cover all kinds of distributed processing. As this is a very ambitious goal, it should not be of great surprise that the ODP viewpoints have different significance as related to various application or system domains.

In this section, the information and computational viewpoints will be considered. These viewpoints are particularly important as they together contain every functional aspect of a system or reference point, such as the definition of information structures and constraints, and identification of computational units. RM-ODP does not prescribe any precise way of achieving a mapping or ensuring consistency between the two viewpoints. In many cases, the mapping between the two viewpoints is non-trivial, and depending on the kind of system or application domain in focus, the role of the two viewpoints and the way they map together will differ. This will be exemplified in the next few paragraphs.

In the application domain of network management, the information viewpoint plays a dominant role. An entire information specification can be associated with a reference point between a managing and a managed system, and it specifies the managed system as perceived from this reference point. Another paper in this issue of *Telektronikk* provides an example of how to use RM-ODP in the network management application domain [7].

On the other hand, consider for example real-time telecom systems. In this application domain, the computational viewpoint will have a dominant role, as numerous communicating components or computational objects are specified. For a discussion of using RM-ODP for the specification of GSM, see Hellan et al. [16]. In their discussion, they show how an information specification can be used to specify constraints related to associations among computational objects as well as how various information elements are associated with various computational objects. In this application domain, the information specification becomes "fragmented", as different parts of the information specification relate to and place constraints on different computational objects.

In the case of network management, the correspondence between an information specification and a computational specification is simple, when considered from a high level. In this case, a computational object can be considered to correspond with an entire system. However, the refinement of such a high level computational object under the constraint of an information specification, into a set of potentially distributed interacting lower level computational objects is non-trivial.

The notion of "an ODP system" is used extensively throughout the RM-ODP specifications. Part 1 states: "A viewpoint is a subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the design of the system". Developing a clever system- or systems architecture is by itself a challenging task, and systems or sub-systems can be related in complex ways. Relationships among viewpoint specifications then become more complex and the one-to-one case can no longer be assumed. Complex relationships among viewpoint specifications have not been addressed by RM-ODP.

In addition to putting the focus on the system, it can also be feasible to identify interworking reference points[5] (IRPs) between systems, and develop precise specifications of these. One challenge not covered by RM-ODP is then how to relate an IRP specification with corresponding specifications of the distributed components realising the IRP. This issue is analogous to refinement of a high level computational object into a set of lower level computational objects.

10.2 Multiparty Interworking Reference Points

The notion of an IRP specification is useful in regard to specification of a binary reference point between two systems such as a managed and a managing system. Such a specification is essentially an information specification[6], and a model-based specification technique [17] is appropriate. However, cases exist where an interaction between two systems may relate to the state space and interactions with one or several other systems, as well. This situation is often characterised by the systems being autonomous, and that each system has their own dedicated view of the state space of each of the other systems.

In this case, the notion of IRP can be used to specify co-ordinated interworking and behaviour among several systems. A multiparty IRP specification can then be considered as a specification of multiparty inter-system interworking. A combined use of a model-based and an interaction-based formal specification technique should be considered for the specification of multiparty IRPs.

Although transient inconsistencies among the systems cannot be avoided, one can avoid permanent inconsistencies. This can be achieved by means of generic consistency preserving mechanisms (transactions).

10.3 Reusing and combining specifications

Reusing and "merging" several generic specifications represents a particular challenge. The generic specifications may need to be customised and profiled by excluding some features and adding other features. In addition, optionality from the generic specification will often be resolved. Features from the various generic specifications may correlate in unexpected ways. The introduction of viewpoint relative specifications complicates these matters. RM-ODP does not address the challenge of reusing and combing viewpoint specifications.

10.4 Flexibility and evolution

Flexibility was identified as one of the objectives of RM-ODP. Support of systems evolution is an integral aspect of flexibility. The relevance of systems evolution will vary from application domain to application domain. However, precise logistics of application versions and corresponding IRP specifications is required to support systems evolution.

RM-ODP Part 2 identifies the notion of epoch as a period of time for which an object displays a particular behaviour. A change of epoch may be associated with a change of the type of the object. In this sense, the notion of epoch can be related to type evolution, versioning and extensibility. However, RM-ODP does not go into any detail regarding the issue of evolution. In particular, the co-ordinated evolution of related viewpoint specifications is a challenging task that needs to be addressed.

11 Conclusion

The RM-ODP standards have provided significant contribution to the progress of distributed processing. The five viewpoints with their corresponding languages, as well as the generic RM-ODP concepts and rules, represent an important base of insight in this field.

However, several challenges related to specification and modelling of distributed processing and systems still remain open. Depending on application domain, rigorous ways of relating information viewpoint specifications and computational viewpoint specifications are needed. Refinement, composition, and evolution of multiple viewpoint specifi-

[5] *RM-ODP states that an interworking reference point is an interaction point. An interaction point is a location where a set of interfaces exists. The notion of interworking reference point (IRP) as adopted in this discussion is assumed not to be constrained by a unit of distribution or a location in space. Thus, the set of interfaces comprising an IRP can be distributed.*

[6] *In [7], the author argues that it is useful to extend such an information specification using the interface construct.*

cations must be addressed. These fundamental relationships among specifications must be developed in order to provide an adequate set of specification tools.

In addition to the RM-ODP standardisation community itself, other arenas will also contribute to the further development of RM-ODP. Significant developments are made by the Object Management Group (OMG), for example through its work on the Unified Modelling Language (UML), and the Business Object Architecture. Other arenas making contributions are various workshops and conferences such as FMOODS[7] and the OOPSLA[8] workshops on Object Oriented Behavioral Semantics. As many of the open issues are related to concepts and rules concerning specification expressiveness, semantics and specification relationships, the just mentioned efforts will also contribute to the further development of formal specification techniques.

[7] *IFIP International Conference on Formal Methods for Open Object-based Distributed Systems* *http://www.cs.ukc.ac.uk/research/netdist/fmoods/*

[8] *ACM Conference on Object-Oriented Programming Systems, Languages and Applications.* *http://www.acm.org/sigplan/oopsla/*

12 References

1 ITU-T. *Information technology : Open distributed processing : Reference Model : Overview.* Geneva, ITU, 1997. (ITU-T Rec. X.901.) (Common text with ISO/IEC.)

2 ITU-T. *Information technology : Open distributed processing : Reference Model : Foundations.* Geneva, ITU, 1995. (ITU-T Rec. X.902.) (Common text with ISO/IEC.)

3 ITU-T. *Information technology : Open distributed processing : Reference Model : Architecture.* Geneva, ITU, 1995. (ITU-T Rec. X.903.) (Common text with ISO/IEC.)

4 ITU-T. *Information technology : Open distributed processing : Reference Model : Architectural Semantics.* Geneva, ITU, 1997. (ITU-T Draft Rec. X.904.) (Common text with ISO/IEC.)

5 Linington, P F. RM-ODP : The Architecture. In: Raymond, K, Armstrong, L (eds.). *(ICODP ’95) Proceedings Third IFIP International Conference on Open Distributed Processing.* London, Chapman & Hall, 1995.

6 Iggulden, D et al. *Architecture and Frameworks, APM.1017.02.* Architecture Projects Management Ltd, UK, 1994.

7 Lønsethagen, H. RM-ODP for Transport Network Modelling : An Example. *Telektronikk,* 94 (1), 55–66, 1998 (this issue).

8 Wegner, P. Dimensions of object-based language design. In: *Proceedings of the OOPSLA’87,* 168–182, 1987.

9 Snyder, A. The Essence of Objects : Concepts and Terms. *IEEE Software,* 1993.

10 OMG. *The Common Object Request Broker : Architecture and Specification, Revision 2.2.* February 1998.

11 Sinnot, R O, Turner, K J. Type Checking in Open Distributed Systems : a Complete Model and its Z Specification. In: Rolia, J et al. *Proceedings of the IFIP/IEEE international conference on Open Distributed Processing and Distributed Platforms (ICODP/ICDP’97), Toronto, Canada, May 1997.* London, Chapman & Hall, 1997.

12 Johannessen, K. Engineering communicating systems. *Telektronikk,* 94 (1), 79–94, 1998 (this issue).

13 Solbakken, H et al. CORBA as an infrastructure for distributed computing and systems integration. *Telektronikk,* 94 (1), 107–118, 1998 (this issue).

14 Sinnott, R O, Turner, K J. Applying formal methods to standard development : the open distributed processing experience. *Computer Standards & Interfaces,* 17, 1995.

15 Milosevic, Z, Bearman M. Towards a new ODP Enterprise Language. In: Rolia, J et al. *Proceedings of the IFIP/IEEE international conference on Open Distributed Processing and Distributed Platforms (ICODP/ ICDP’97), Toronto, Canada, May 1997.* London, Chapman & Hall, 1997.

16 Hellan, J K et al. On the Applicability of an ODP-compatible Framework to the Specification of GSM. In: *The Fifth TINA Conference, Melbourne, Australia, Feb. 1995.*

17 Goldsack, S J, Kent, S J H (eds). *Formal Methods and Object Technology.* Berlin, Springer, 1996.

Håkon Lønsethagen is Research Scientist at Telenor R&D, Kjeller, Network and Service Management Platform Unit. Currently, his focus is access network management and ATM transport network management. His research interests are systems architecture, systems evolution and specification techniques.

e-mail:
hakon.lonsethagen@fou.telenor.no

RM-ODP for Transport Network Modelling – An Example

HÅKON LØNSETHAGEN

The following provides an example to illustrate one way to use RM-ODP[1]. The example is taken from the network management application domain. This approach of using RM-ODP has been developed by ITU-T SG15[2]. Comments will be made throughout the example both with respect to this particular way of using RM-ODP as well as to what the gains and advantages are of using RM-ODP in this application area. A few comments will also be made with respect to the TINA-C[3] work and their use of RM-ODP.

1 Introduction

This example is based on the "Simple Subnetwork Connection Configuration" management application as defined by ITU-T SG15 in Recommendations G.852.1, G.853.2 and G.854.1 [6, 7, 8]. This represents one management application of several, where many are still under development. SG15 has developed a methodology based on RM-ODP suitable for the purpose of transport network modelling which is documented in a separate recommendation G.851.1 [9]. With respect to the notion of logical layered architecture as suggested by TMN (M.3010 – [10]) these specifications address the network management layer.

The example will select and present parts of these recommendations to demonstrate the main features of each RM-ODP viewpoint and corresponding language. Discussion of detailed issues of these languages will not be made. Comments will be made at the end of the example raising a few issues related to this approach.

The SG15 specification effort is based on a generic functional architecture of digital transport networks – G.805 [11]. G.805 defines architectural concepts related to functional aspects of transport network resources, such as topological components, transport entities and transport processing functions.

In order to provide some background information, Figure 1 introduces the network architectural entities used in this example. Some of the definitions are "simplified" to avoid the need of presenting the entire G.805 model.

- *topological component:* An architectural component used to describe the transport network in terms of the topological relationships between sets of points within the same layer network.
- *transport entity:* An architectural component which transfers information between its inputs and outputs within a layer network.
- *characteristic information:* A communication signal and specific transport format which is transferred on a transport entity.
- *layer network:* A topological component that includes both transport entities and transport processing functions that describe the generation, transport and termination of a particular characteristic information.
- *port:* It consists of a pair of associated input/output (sink/source) of a transport entity.
- *subnetwork:* A topological component used to effect routing of a specific characteristic information within a specific layer network.
- *subnetwork connection:* A transport entity that transfers information across a subnetwork, it is formed by the association of ports on the boundary of the subnetwork.

Figure 1 Example G.805 definitions

2 Enterprise Viewpoint

In general terms, the enterprise viewpoint is concerned with the business activities of the specified system by focusing on purpose, scope, and policies. The following statement reflects the role of the enterprise viewpoint as considered by SG15. "The enterprise viewpoint is included in the network model in order to:

- document the use of the various portions of the model based on *communities* of common functions (applications); and
- provide a way of specifying the requirements upon which the model is defined."

A community specifies the scope of the specific management application in focus, and comprises a set of roles, a set of actions, and a set of policies to satisfy the common objective, or contract, that is shared between the roles. A community description represents a set of potential community contract instances, each of which reflects a particular selection of service features available, and each based on negotiation. Thus, a contract represents a service with an associated quality that is offered in a client/provider relationship by the provider to its client.

Figure 2 shows the example in the left column. Comments are provided in the right column to explain the most important concepts used in the corresponding description. These comments do not provide an exhaustive presentation of relevant issues with this respect. The methodology document itself (G.851.1 – [9]) should be conferred for a more complete presentation. Comments may also be located in-line with the example. Italics will be used for this purpose. Comments in italics in the right column, reflect related comments by the author. This scheme will be used for the other examples as well.

The rationale behind the enterprise viewpoint is to provide a de-coupling of

[1] *Reference Model of Open Distributed Processing [1–4]. See another paper in this issue of* Telektronikk *for an introduction to RM-ODP.*

[2] *The work on network level management of transmission systems has been transferred to SG4 Question 18 for the study period 1997–2000.*

[3] *TINA-C: Telecommunications Information Networking Architecture Consortium. RM-ODP is one of the foundations of TINA [5]. http://www.tinac.com/*

COMMUNITY sscc “Simple Subnetwork Connection Configuration”	A community comprises six elements: Purpose, Role, Policy, Action (with action policies), Activity and Contract
1 PURPOSE “The objective of the community is to configure point-to-point subnetwork connections between pairs of ports, which have been previously populated on the boundary of a subnetwork.”	This clause introduces the purpose and objective of the community as a whole.
2 ROLE	This clause introduces all the roles in the community.
caller “This role reflects the client of the actions defined in this community. One and only one caller role occurrence may exist in the community.”	A caller role represents the behaviour of an enterprise object that defines the service requests of a given service.
provider “This role reflects the server of the actions defined in this community. One and only one provider role occurrence must exist in the community.”	A provider role represents the behaviour of an enterprise object that performs the service requests of a given service.
port “This role reflects the G.805 port resource used in the actions defined in this community. Zero or more port role occurrences may exist in the community.” **sn** “This role reflects the G.805 subnetwork resource used in the actions defined in this community. One and only one sn role occurrence must exist in the community.” **snc** “This role reflects the G.805 subnetwork connection resource used in the actions defined in this community. Zero or more snc role occurrences may exist in the community.”	Other roles of a service represent behaviour of enterprise objects, reflecting the involvement of resources in the context of this service.
3 POLICY OBLIGATION OBLG_1 “The provider shall support such viewing of the resource properties and relationships which have been identified and allowed in the service contract with the caller.”	These statements give the overall policies associated with the community.
4 ACTION **4.1 sscc1 “Setup Point-to-Point SNC”** “This action sets up a point-to-point subnetwork connection between two ports on the same subnetwork.”	The ACTION clause provides a list of actions which support the purpose of the community.
ACTION_POLICY OBLIGATION OBLG_1 “This action must be provided in respect to all the action policies or it fails. In the event the action fails, the provider shall report to the caller which action policy has been violated.” OBLIGATION OBLG_2 “The caller shall identify two ports which must be part of the community.”	These policies are associated with the corresponding action, and should state the role and information involved. It is the intent not to be prescriptive about the information in the enterprise viewpoint.

Figure 2 Example enterprise

objectives and requirements of an ODP system from its realisation. The above example illustrates how this aim is accomplished. Furthermore, note that the enterprise viewpoint descriptions are application-driven. An important intention of this approach is not to be prescriptive about the information elements introduced.

The enterprise viewpoint does not identify computational interfaces. However, the notion of a community implies an abstract, high level “interface” between roles – in this case the caller and the provider role. Whenever such an interface is described, one must make some design decisions about information and functionality associated with this abstract interface. This becomes clear if one tries to use a formal language for the enterprise specification. Thus, the position of an enterprise specification relative to an information and a corresponding computational specification only becomes a matter of level of detail or specialisation of the specifications.

OBLIGATION OBLG_3
"In case of service establishment, the provider shall inform the caller of the subnetwork connection identifier that is unique in the community for the duration of the subnetwork connection."

PERMISSION PERM_1
"The caller may provide a user identifier for the requested subnetwork connection."

OBLIGATION OBLG_4
"If PERM_1 is part of the contracted service, then the provider shall use the user identifier, if provided, as a unique subnetwork connection identifier when communicating with the caller."

OBLIGATION OBLG_5
"If PERM_1 is part of the contracted service and if the user identifier is not unique in the provider context, then the provider shall reject the user identifier and inform the caller of the rejection."

PERMISSION PERM_2
"The caller may specify the characteristics of the requested transport service (e.g. bandwidth, directionality, route selection criteria, availability, etc.). These are contract and technology specific."

PROHIBITION PROH_1
"The provider shall not satisfy the request if one or both the ports are already used in a subnetwork connection."

4.2 sscc2 "Release Point-to-Point SNC"
"This action is used to release a simple point-to-point subnetwork connection."

ACTION_POLICY
OBLIGATION OBLG_1
"This action must be provided in respect to all the action policies or it fails. In the event the action fails, the provider shall report to the caller which action policy has been violated."

OBLIGATION OBLG_2
"The caller shall uniquely identify an existing subnetwork connection via its subnetwork connection identifier."

OBLIGATION OBLG_3
"The provider shall inform the caller of the subnetwork connection identifier of the subnetwork connection which has been released."

5 ACTIVITY
"None."

The ACTIVITY clause is used in the case where the client has to address several related (ordered) actions to the provider as part of a service feature.

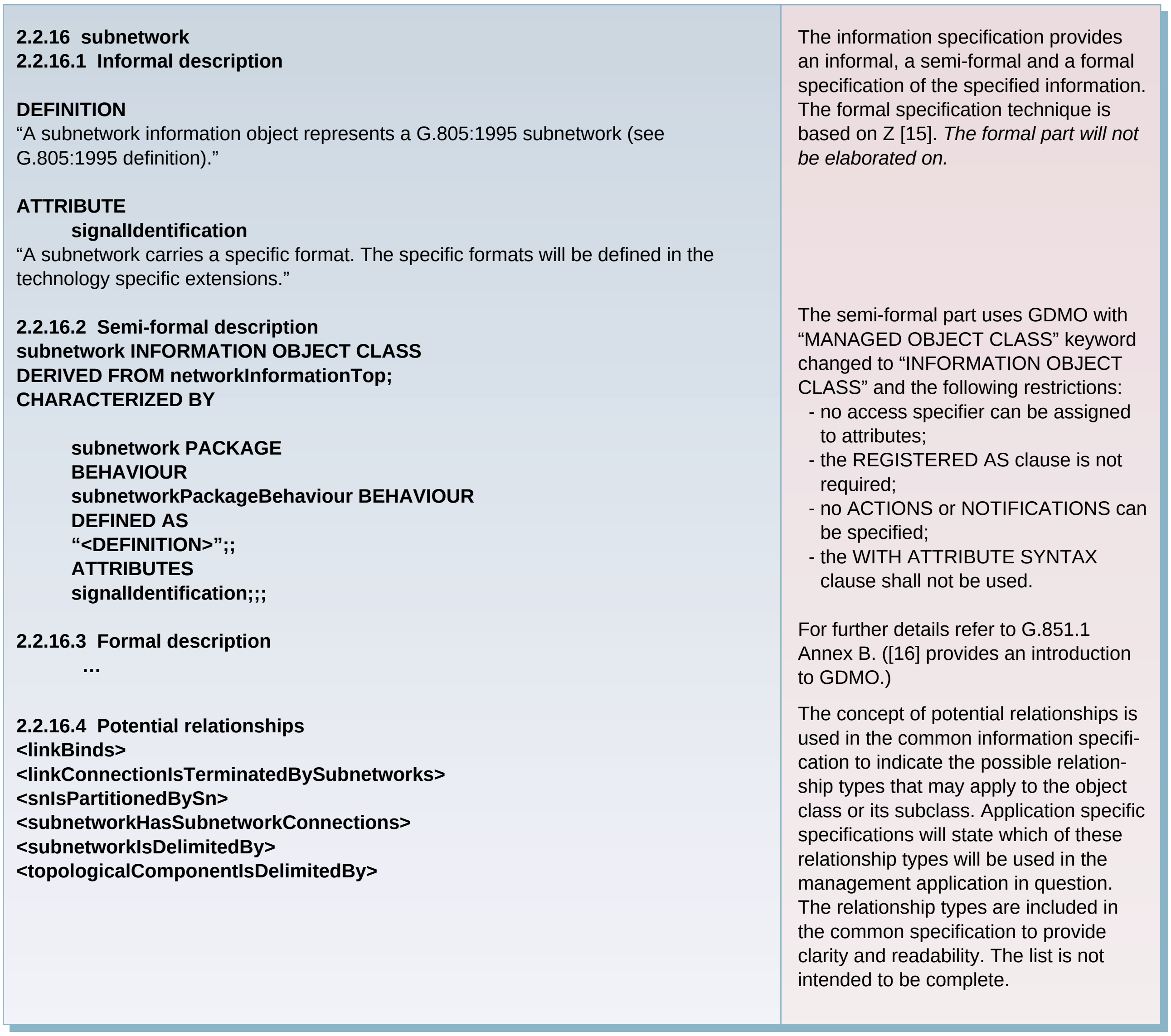
6 CONTRACT
"Service features subject to negotiation as part of the service contract shall include:
- degree of visibility of resource (their properties and relationships); and
- other features which are for further study."

A contract is the result of negotiation reflecting the agreed selection from the set of service features provided by the service provider.

viewpoint specification

The above remark is valid considering the functional aspects of an enterprise specification. In addition, an enterprise specification also provides a means of capturing high level requirements and objectives associated with the computing infrastructure and other non-functional aspects. Infrastructure and other non-functional aspects have not been focused in the SG15 work.

TINA-C has adopted a somewhat different approach with respect to the enterprise viewpoint. A particular enterprise viewpoint description template has not been developed. However, various TINA documents play the role of an enterprise description as they describe business models where stakeholders are identified representing actors or agents of the ODP (TINA) system. The associated objectives and requirements are identified at a general service independent level, and non-functional aspects are also considered to some degree. The TINA business model and requirements documents



2.2.16 subnetwork
2.2.16.1 Informal description

DEFINITION
“A subnetwork information object represents a G.805:1995 subnetwork (see G.805:1995 definition).”

ATTRIBUTE
signalIdentification
“A subnetwork carries a specific format. The specific formats will be defined in the technology specific extensions.”

```
2.2.16.2 Semi-formal description
subnetwork INFORMATION OBJECT CLASS
DERIVED FROM networkInformationTop;
CHARACTERIZED BY

        subnetwork PACKAGE
        BEHAVIOUR
        subnetworkPackageBehaviour BEHAVIOUR
        DEFINED AS
        “<DEFINITION>”;;
        ATTRIBUTES
        signalIdentification;;;

2.2.16.3 Formal description
         ...

2.2.16.4 Potential relationships
<linkBinds>
<linkConnectionIsTerminatedBySubnetworks>
<snIsPartitionedBySn>
<subnetworkHasSubnetworkConnections>
<subnetworkIsDelimitedBy>
<topologicalComponentIsDelimitedBy>
```

The information specification provides an informal, a semi-formal and a formal specification of the specified information. The formal specification technique is based on Z [15]. *The formal part will not be elaborated on.*

The semi-formal part uses GDMO with “MANAGED OBJECT CLASS” keyword changed to “INFORMATION OBJECT CLASS” and the following restrictions:
- no access specifier can be assigned to attributes;
- the REGISTERED AS clause is not required;
- no ACTIONS or NOTIFICATIONS can be specified;
- the WITH ATTRIBUTE SYNTAX clause shall not be used.

For further details refer to G.851.1 Annex B. ([16] provides an introduction to GDMO.)

The concept of potential relationships is used in the common information specification to indicate the possible relationship types that may apply to the object class or its subclass. Application specific specifications will state which of these relationship types will be used in the management application in question. The relationship types are included in the common specification to provide clarity and readability. The list is not intended to be complete.

Figure 3 Example common information specification

thus set the stage for further TINA information and computational specifications.

From the above discussion, it is concluded that requirements capture is the foremost objective of the enterprise viewpoint.

3 Information Viewpoint

The information viewpoint is concerned with the information that needs to be stored and processed, and the semantics of the information.

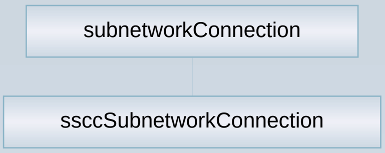
G.853.2 “Subnetwork connection management information viewpoint” [7] represents the information specification related to the enterprise specification in G.852.1. However, G.853.2 is in turn based on a management application independent common information specification G.853.1 “Common elements of the information viewpoint for the management of a transport network” [12]. The common information specification is a management application independent description of information objects and relationship classes, representing G.805 transport network resources. G.853.1 then provides a basis for application specific information specifications. The latter can extend the common classes by specifying additional subclasses. Additional relationship classes can also be specified in application specific information specifications.

The specification templates used are modifications of the GDMO [13] and GRM [14] templates respectively. The modification of GDMO will be explained along with the example presented. GRM is extended with the opportunity of providing more than just one object class label in the “COMPATIBLE WITH” clause. By this opportunity, one avoids the need of writing down “duplicate” relationships.

Figure 3 shows an example selected from G.853.1, the common information specification. It is provided to show one example of an information object class definition specified in the common information specification.

An example illustrating the full structure of an information specification is provided in Figure 4. This example is selected from G.853.2 Annex A: “Information viewpoint for the simple subnetwork connection management”. Thus, this is an example of a management application specific information specification. The example shows selected parts of G.853.2

1 Diagrams of information objects and relationship classes



Inheritance diagram

2 Label references

(Label references are used to provide local names of information entities specified in other documents. For example:)

Full label reference	Local label reference
<“Rec.G.853.1”,INFORMATION_OBJECT: subnetwork>	<subnetwork>

3 Import

<“Rec.G.853.1”,INFORMATION_RELATIONSHIP: subnetworkIsDelimitedBy>
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,PURPOSE>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_1>

<“Rec.G.853.1”,INFORMATION_RELATIONSHIP: subnetworkConnectionIsTerminatedByPointToPoint>
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,PURPOSE>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_2>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:PROH_1>

<“Rec.G.853.1”,INFORMATION_RELATIONSHIP: subnetworkHasSubnetworkConnections>
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,PURPOSE>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_3>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_4>

<“Rec.G.853.1”,ATTRIBUTE:userLabel>
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_3>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:PERM_1>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_4>
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,OBLIGATION:OBLG_2>

4 Information object class definition

4.1 ssccSubnetwork

This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ROLE:sn>

4.1.1 Informal description

DEFINITION
  “This object class is derived from subnetwork.”
RELATIONSHIP
  “<subnetworkIsDelimitedBy>”,
  “<subnetworkHasSubnetworkConnections>”

4.2 ssccSubnetworkConnection

This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ROLE:snc>

4.2.1 Informal description

DEFINITION
  “This object class is derived from subnetworkConnection.”
ATTRIBUTE
  userLabel
    “This attribute is used to identify the ssccSubnetworkConnection”
RELATIONSHIP
  “<subnetworkIsDelimitedBy>”,
  “<subnetworkConnectionIsTerminatedByPointToPoint>”,
  “<subnetworkConnectionHasTSC>”,
  “<subnetworkHasSubnetworkConnection>”

Diagrams are provided for readability. (This example does not show all diagrams of Annex A.)

(The ‘sscc’ prefix is used throughout the Recommendation and is an abbreviation for simple subnetwork connection configuration.)

(Label references are also used in other clauses, as will become evident below. The label reference structure and syntax will not be covered in any more detail here.)

The Import clause provides a list of information entities or concepts which have been specified in other documents. Corresponding to each information concept a relevant part of an enterprise specification is referenced. This approach illustrates how correspondence between the enterprise and the information viewpoints is provided.

(In the following only the informal descriptions will be presented.)

The mandatory relationships for this management application are listed in the RELATIONSHIP section. The relationships are selected from the list of potential relationships of the common information specification.

Additional relationships may be defined in application specific specifications. *The relationship* <subnetworkConnectionHasTSC> *is an example of this possibility. The actual definition will be done below.*

Figure 4 Example information viewpoint specification (panel 1 of 3)

4.3 ssccSubneworkTPBidirectional
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ROLE:port>

(TP – Termination Point)

4.3.1 Informal description
DEFINITION
“This object class is derived from ssccSubnetworkTPSink and ssccSubnetworkTPSource.”

4.4 ssccSubnetworkTPSink
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ROLE:port>

4.4.1 Informal description
DEFINITION
“This object class is derived from subnetworkTPSink .”
RELATIONSHIP
“<subnetworkIsDelimitedBy>”,
“<subnetworkConnectionIsTerminatedByPointToPoint>”

(From G.853.1 [12]: The subnetworkTP-Sink information object is an abstraction that represents the potential termination of a transport entity and the associated unidirectional port. It also represents the potential for connection across sub-networks. Correspondingly, the subnetworkTPSource represents the potential origin of a transport entity.)

4.5 ssccSubnetworkTPSource
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ROLE:port>

4.5.1 Informal description
DEFINITION
“This object class is derived from subnetworkTPSource .”
RELATIONSHIP
“<subnetworkIsDelimitedBy>”,
“<subnetworkConnectionIsTerminatedByPointToPoint>”

4.6 serviceCharacteristics
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,PERMISSION:PERM_2>

4.6.1 Informal description
DEFINITION
“This object class reflects all the characteristics associated with the requested quality of the transport service relevant to the subnetwork connection establishment. This object class will be refined due to the (transport) technological dependent characteristics.”
RELATIONSHIP
“<subnetworkConnectionHasTSC>”

5 Information relationship definitions
5.1 subnetworkConnectionHasTSC
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1,PERMISSION:PERM_2>

5.1.1 Informal description
DEFINITION
“The subnetworkConnectionHasTSC relationship type describes the association between a subnetwork connection and the related quality of Transport Service Characteristics.”
ROLE
transportQualified
“Played by instances of the ssccSubnetworkConnection information object class and subclasses”
transportQualifier
“Played by an instance of the serviceCharacteristics object class.”
INVARIANT
inv_1
“Several objects playing the transportQualified role may be involved in the relationship.”
Inv_2
“Only one object playing the transportQualifier role may be involved in the relationship.”

(These roles are relationship roles [14] and should not be confused with roles as identified in the enterprise viewpoint.)

Figure 4 Example information viewpoint specification (panel 2 of 3)

6 Static schemas
6.1 ssccNotConnected
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1, OBLIGATION:OBLG_2>,
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1, PROHIBITION:PROH_1>,
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc2, OBLIGATION:OBLG_2>

6.1.1 Informal description
DEFINITION
> **“The ssccNotConnected schema defines a schema type with two non-connected subnetworkTP information object subtype candidates to the point-to-point connection management service.”**

ROLE
> **involvedSubnetwork**
> **“Played by an instance of the ssccSubnetwork information object type or sybtype.”**
> **potentialAEnd**
> **“Played by an instance of the ssccSubnetworkTPSink, ssccSubnetworkTPSource or ssccSubnetworkTPBidirectional object types or subtypes.”**
> **potentialZEnd**
> **“Played by an instance of the ssccSubnetworkTPSink, ssccSubnetworkTPSource or ssccSubnetworkTPBidirectional object types or subtypes.”**

INVARIANT
> **INV_1**
> **“The object playing the potentialAEnd and potentialZEnd roles are involved in an instance of the subnetworkIsDelimitedBy relationship type with the object playing the role involvedSubnetwork.”**
> **INV_2**
> **“The object playing the potentialAEnd role is not involved in any instance of the subnetworkConnectionIsTerminatedByPointToPoint relationship type or subtype.”**
> **INV_3**
> **“The object playing the potentialZEnd role is not involved in any instance of the subnetworkConnectionIsTerminatedByPointToPoint relationship type or subtype”**

6.2 ssccConnected
…

7 Dynamic Schemas

7.1 ssccNotConnected_ssccConnected
This information concept is related to the following enterprise entities:
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1 >.

7.1.1 Informal description
DEFINITIOIN
> **“This dynamic schema expresses the transition of two non-connected extremities toward two connected extremities ”**

PRE-CONDITIONS
> **<ssccNotConnected>;**

POST-CONDITIONS
> **<ssccConnected>;**

7.2 ssccConnected_ssccNotConnected
…

8 Attributes
> **“None”**

A static schema is used to define a state of a system (of this kind) at an instance of time. This global/generic (compound) state involves one or more attributes of one or more information objects. A set of static schemata is used to describe the pre-condition of a global system state transition. Likewise, a set of static schemata is used to describe the post-condition of a transition. It is only necessary to specify the static schemas that are of interest.

The DEFINITION clause provides the overall semantic of the global state.

The ROLE clause identifies all the static schema roles used in descriptions of invariants (see below), and which object classes that can play each of these roles.

The INVARIANT clause provides the applicable attribute value constraints. These constraints can be within an object or between objects through relationship constraints.

A dynamic schema identifies a particular transition between compound states (described by static schemata). Transitions (dynamic schemata) will be referenced and further described in a corresponding computational specification.

Attribute types will typically be defined in the common information specification for reuse by other information specifications. The semi-formal description of attribute types are based on the ATTRIBUTE template of GDMO excluding the WITH ATTRIBUTE SYNTAX clause.

Figure 4 Example information viewpoint specification (panel 3 of 3)

0 Computational interfaces to satisfy simple subnetwork connection configuration community enterprise requirements

The simple subnetwork connection configuration enterprise requirements are met by the interfaces specified in this Annex.

1 Label references

The following information relationships, static schema, and ASN.1 productions are referenced in this Annex:

Fully qualified label reference	Local reference used
<“Rec.G.853.1”,INFORMATION_RELATIONSHIP: subnetworkIsDelimitedBy>	<subnetworkIsDelimitedBy>
…	
<“Rec.G.853.2”,STATIC_SCHEMA:ssccNotConnected>	<ssccNotConnected>
Fully qualified ASN.1 production reference	**Local reference used**
“M.3100 : 199x : ASN1DefinedTypesModule”::Failed	Failed
“M.3100 : 199x : ASN1DefinedTypesModule”::Directionality	Directionality
“M.3100 : 199x : ASN1DefinedTypesModule”::UserLabel	UserLabel

(Example label references.)

2 simple SNC performer interface

The simple subnetwork performer manages the set-up and release of subnetwork connections. The simple SNC performer interface is required to satisfy the enterprise requirements stated in:

<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc1 >,
<“Rec.G.852.1”,COMMUNITY:sscc,ACTION:sscc2 >.

This interface provides basic connection set-up functionality. The operation ssccSetupSubnetworkConnection sets up a subnetwork connection, and the operation ssccReleaseSubnetworkConnection removes the subnetwork connection.

```
COMPUTATIONAL_INTERFACE simpleSncPerformerIfce {
        OPERATION    <ssccSetupSubnetworkConnection>;
                     <ssccReleaseSubnetworkConnection >;
}
```

2.1 sscc set-up SNC

This operation sets up a simple subnetwork connection between a single A-End snTP or nTP, and a single Z-end snTP or nTP.

(snTP – subnetwork termination point. nTP – network termination point. nTP is a generalisation of networkTTP and networkCTP. TTP – trail termination point. CTP – connection termination point. For further explanations, conf. G.853.1 [12].)

```
OPERATION       ssccSetupSubnetworkConnection {
  INPUT_PARAMETERS
        subnetwork : SubnetworkId ::= (ssccSnIfce)
– If the performer is associated with only one subnetwork, the subnetwork parameter of this operation is
redundant and may be removed as an engineering optimization.
        snpa : SnTPId ::= (snTPIfce);
        snpz : SnTPId ::= (snTPIfce);
        dir : Directionality;
        suppliedUserLabel : UserLabel;
        serviceCharacteristics : CharacteristicsId ::=
                    (serviceCharacteristicsIfce);
```

In the parameter declaration to the left, snpa is the name of the formal parameter, SnTPId is the name of the parameter type, and “::= (snTPIfce)” declares the parameter type to conform to the snTPIfce interface signature type. The indirect type assignment is introduced to allow mapping from communication domain independent interface types to communication domain dependent interface types.

Figure 5 Example computational

Annex A to illustrate how this specification in turn corresponds to a corresponding computational specification.

The example illustrates how elements of an enterprise specification are referenced from an information specification, thus providing correspondence between the two viewpoints. Furthermore, one can observe that information entities have been identified and described (prescribed), possibly by using a formal language. Note that an information specification defines one (logically centralised) state space and its constraints of an abstract system. Although possible transitions of that state space of such a system can be identified in an informa-

```
  OUTPUT_PARAMETERS
        newSNC : SNCId ::= (sncIfce);
        agreedUserLabel : UserLabel;

  RAISED_EXCEPTIONS
        incorrectSubnetworkTerminationPoints : SEQUENCE OF SnTPId;
        subnetworkTerminationPointsConnected : SEQUENCE OF SnTPId;
        invalidTransportServiceCharacteristics : NULL;
        failure : Failed;
        wrongDirectionality : Directionality;
        userLabelInUse : UserLabel;

BEHAVIOUR
  INFORMAL
        ...
  SEMI_FORMAL
        PARAMETER_MATCHING
        subnetwork : <ssccNotConnected, ROLE:involvedSubnetwork> AND
                      <ssccConnected, ROLE:involvedSubnetwork> ;
        snpa : <ssccNotConnected, ROLE:potentialAEnd> AND
                      <ssccConnected, ROLE:connectedAEnd> ;
        snpz : <ssccNotConnected, ROLE:potentialZEnd> AND
                      <ssccConnected, ROLE:connectedZEnd> ;
        dir : <ssccConnected, ROLE:involvedSubnetwork, ATTRIBUTE: directionality> ;
        newSCN : <ssccConnected, ROLE:involvedPointToPointSubnetworkConnection>;
        suppliedUserLabel : <ssccConnected, ROLE:involvedSubnetwork,
                      ATTRIBUTE: userLabel> OR <> ; – – The user does not have to supply a
                                                     user label value
        agreedUserLabel : <ssccConnected, ROLE:involvedSubnetwork,
                      ATTRIBUTE: userLabel>  ;
        serviceCharacteristics : <ssccConnected, ROLE:involvedServiceCharacteristics> ;

        PRE_CONDITIONS  <ssccNotConnected> ;
        POST_CONDITIONS  <ssccConnected> ;

        EXCEPTIONS
        IF PRE_CONDITION <inv_1> NOT_VERIFIED RAISE_EXEPTION
                      incorrectSubnetworkTerminationPoints ;
        IF PRE_CONDITION <inv_2> NOT_VERIFIED RAISE_EXEPTION
                      subnetworkTerminationPointsConnected ;
        IF PRE_CONDITION <inv_3> NOT_VERIFIED RAISE_EXEPTION
                      subnetworkTerminationPointsConnected ;
        IF POST_CONDITION <inv_1> NOT_VERIFIED RAISE_EXEPTION failure ;
        IF POST_CONDITION <inv_2> NOT_VERIFIED RAISE_EXEPTION failure ;
        IF POST_CONDITION <inv_3> NOT_VERIFIED RAISE_EXEPTION failure ;
        IF POST_CONDITION <inv_4> NOT_VERIFIED RAISE_EXEPTION userLabelInUse ;
  ;
}

2.2  sscc release SNC
     ...
```

(While input parameters are provided in the invocation by the caller, the output parameters are provided in the corresponding termination by the provider.)

A parameter matching clause specifies the set of information objects or attributes that are intended to be bound to the parameter. It specifies an information object, either directly or as a ROLE played in an information relationship, possibly via a static schema role as in this example, or an attribute value of an information object.

(<inv_x> refers to invariants of the corresponding static schema. See Figure 4, panel 3.)

viewpoint specification

tion specification, no decision has been made as to how the transitions relate to interfaces of a corresponding computational or engineering specification. Moreover, specific operations and their parameters are not part of this specification.

TINA has adopted a similar approach of using GDMO in combination with GRM for the information viewpoint [17]. However, TINA allows specification of operations as a part of the information viewpoint.

4 Computational Viewpoint

The computational viewpoint is concerned with distribution by addressing functional decomposition into computational objects which interact at interfaces.

This section will follow up the two previous ones, and shows an example where computational interfaces for the simple subnetwork connection configuration management application are defined. The example is based on selected parts of G.854.1 "Computational interfaces for basic transport network model" [8]. Although SG15 has developed a template for specifying computational objects, this has not been used in G.854.1. It has been stated that computational objects are only defined for an application if the required interactions between the interfaces *(i.e. between the engineering objects realising the interfaces)* need to be standardised.

The specification approach makes a distinction between communication domain-independent computational specifications and communication domain-dependent computational specifications. The latter being related to a communication domain like CORBA/IIOP [18], OSI Systems Management/CMIP [19, 20], etc. The difference between the two is that the first only uses ASN.1 [21] abstract syntax[4], while the latter uses a particular syntax which will be used in the engineering realisation as well. In addition, a communication domain dependent specification must introduce a common "top" interface type from which the other interfaces are derived. For example for the GDMO/CMISE [13, 22] engineering style, this would resemble the GDMO "top" managed object class [23]. Figure 5 shows a communication domain-independent specification case.

In this example, we have seen how interfaces and operations have been introduced as well as associated operation exceptions. Although not demonstrated, additional sequencing constraints regarding operation sequences may be added in this viewpoint. The example has demonstrated how parameter values are associated with information entities expressed in an information specification. Behavioural constraints have been expressed by referencing static schemata of the information specification.

TINA has also developed a specification language for the computational viewpoint [24, 25]. This is a superset of the CORBA IDL [18], allowing specification of computational objects. The TINA approach does not provide a "parameter matching" facility for stating correspondence with an information specification.

5 Engineering Viewpoint

The engineering viewpoint is concerned with the mechanisms and functions supporting distribution of the computational objects and their interaction.

The focus of the engineering viewpoint, as interpreted by SG15, is the infrastructure technology specific specifications of interfaces and basic engineering objects and how these are grouped into clusters. Non-functional requirements from enterprise specifications must be considered and accommodated when making decisions about basic engineering objects and how they are grouped into clusters and capsules.

Thus far, SG15 has not prescribed any specific engineering specifications. However, their methodology presented in recommendation G.851.1 indicates a way of mapping from the information and computational specifications to GDMO, which in turn can be implemented using CMISE/CMIP. Mappings to other infrastructure specific specifications are for further study.

It is interesting to note the difference in the way SG15 and TINA-C interpret what is a computational object vs. what is a basic engineering object. SG15 considers CORBA IDL [18] objects and GDMO Managed Objects as engineering objects, while TINA-C considers their ODL language [25] (a super set of CORBA IDL) as a computational language.

6 Comments

The above examples have given an indication of how the RM-ODP can be applied within one specific application domain, that is, the field of network management. In this field, the modelling and specification of information representing the resources to be managed is of utmost importance. This is reflected by the importance of the information viewpoint and the common information specification G.853.1 [12]. The essence of the distributed management system is described in this viewpoint, the information to be manipulated and its semantics, and relationships and constraints, static as well as dynamic.

A thorough assessment of the RM-ODP-based approach developed by SG15 as compared with other approaches will not be undertaken here. However, a few comments are made, comparing the SG15-approach with the more traditional OSI Systems Management [19] approach.

The following can be considered as main contributions of the SG15 approach as compared with the OSI Systems Management approach:

- The introduction and use of the enterprise viewpoint for documenting requirements, policies and scope.
- Enabling information specifications without any coupling to distribution, centralisation or implementation.
- Enabling specification of computational interfaces and a mechanism of referencing elements of a corresponding information specification, thus mapping behaviour and constraints specified in the information viewpoint to the computational viewpoint.

To illustrate the difference between this approach and the OSI Management approach, Figure 6 is provided. Figure 6 (a) shows the OSI Management approach. In the figure, the notion of interworking reference point (IRP)[5] is introduced. The concept is used here to designate the interworking between two systems. In the OSI Management case, an IRP specification will correspond to a management information specification using GDMO.

In the OSI Management approach, it is assumed that the communication between the managing and the managed systems is provided by the MIS-user processes in the manager and the agent role respectively. This solution does not support the notion of multiple computational interfaces associated with a MIS-user or a managed system. This results in a centralised solution for the management interactions.

[4] *Only a subset of ASN.1 is allowed, to provide easier translation to other syntaxes.*

[5] *RM-ODP states that an interworking reference point is an interaction point. An interaction point is a location where a set of interfaces exists. The notion of interworking reference point (IRP) as adopted in this discussion is assumed not to be constrained by a unit of distribution or a location in space. Thus, the set of interfaces comprising an IRP can be distributed.*

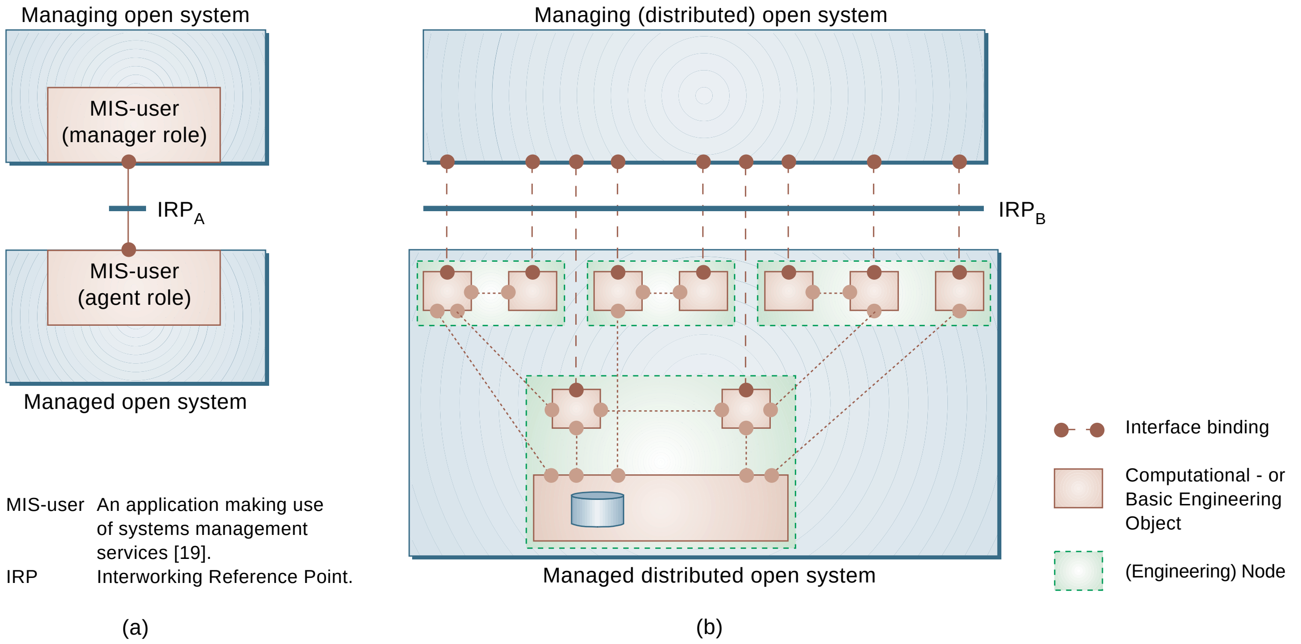

Managing open system
MIS-user (manager role)
IRP_A
MIS-user (agent role)
Managed open system
MIS-user An application making use of systems management services [19].
IRP Interworking Reference Point.
(a)
Managing (distributed) open system
IRP_B
Managed distributed open system
(b)
Interface binding
Computational - or Basic Engineering Object
(Engineering) Node

Figure 6 OSI Management vs. distributed management

The goal of the SG15 approach has been, among others, to relieve this constraint. Instead of assuming interactions with an agent being constrained to one location, the SG15 approach is assuming a "distributed agent", facilitated by a set of possibly distributed computational/engineering interfaces. This is illustrated in Figure 6 (b), where IRP_B includes a set of associated interfaces (interface bindings). The managed system (or more precisely, the agent process) does not have to be located in one node any longer (nor be considered as an entity at all).

At an abstract level, the managed as well as the managing distributed system, can be considered as a high level computational object. The encapsulated state space of this computational object can be specified as perceived at one of its supporting IRPs (e.g. IRP_B), and the behaviour of interactions on an IRP can be specified in terms of changes to this state space. In this way, an IRP is only one view of the state space encapsulated by the computational object (the managed distributed system). The figure further illustrates a sub-structure of such a computational object. This sub-structure is just an example, as there exist numerous ways of realising such a distributed system (computational object). The internal structure of this high level computational object will be hidden from the specification of the IRP.

Extending information specifications with interfaces

The SG15 approach, by the combination of information and computational interface specifications, provides a means of stating IRP specifications of the kind illustrated in Figure 6 (b). However, as the above example has shown, one facet of the SG15 information specification is that it does not identify any operations or interfaces. These are instead identified in the computational viewpoint. However, it is a rather elaborate task to relate a computational specification to the information specification. Accordingly, it is also a challenging task for the reader to comprehend and relate the separate specifications.

Therefore, it could be a good idea to extend information specifications by allowing the notion of interfaces to be used with such specifications. The RM-ODP definition of the information viewpoint and language does not preclude the use of interfaces within the information viewpoint. A mechanism allowing specification of sequencing constraints among operations may also be useful.

The issue of extending information specifications with interfaces is not exclusively relevant to the field of network management. As long as it is appropriate to use a model-based specification technique [26] (i.e. the specification technique assumed for the information language), and one wants to specify a system from the perspective of an IRP, it can be useful to extend such an IRP-related information specification with interfaces. In such a case, an interface instance has a natural correspondence with the lifetime of an information object (instance) or the system instance itself. Due to the correspondence between interfaces and information objects, it seems natural to develop an interface construct to be used with the information language. Pre- and post-conditions related to the existence of information objects will then also naturally apply to the existence of corresponding interface instances, as well.

An information specification with interfaces and sequencing constraints can be considered as a refinement of the corresponding information specification without these constructs. By enabling this refinement step within one language, the refinement task will be easier and more manageable, particularly due to improved readability. IRP specifications, or information specifications with interfaces, can be considered as a merge of the information and the computational viewpoints. Furthermore, it will be possible to compile such an information specification and generate stubs and skeletons for engineering objects according to the chosen infrastructure technology.

The formal specification language Z or any object-oriented extensions of Z [27], extended with an interface construct and a means of expressing sequencing constraints, seems to be an attractive language for the specification of IRPs. By using a formal language, the support for various consistency checking can be taken advantage of.

To conclude this section, the issue of generic interface is addressed. Although not shown in the above example, the computational specification G.854.1 includes several interface specifications used for reading (querying) information. Often, significant specification effort is needed to allow sufficient capabilities for querying information. A more efficient approach as seen from a specification perspective, is to provide a generic interface allowing queries based on a query language. This way, query types do then not need to be specified in advance, and may be dynamically developed and invoked on the interface. However, non-functional requirements associated with the use of this general purpose interface are likely to be needed.

7 References

1 ITU-T. *Information technology : Open distributed processing : Reference Model : Overview.* Geneva, ITU, 1997. (ITU-T Rec. X.901.) (Common text with ISO/IEC.)

2 ITU-T. *Information technology : Open distributed processing : Reference Model : Foundations.* Geneva, ITU, 1995. (ITU-T Rec. X.902.) (Common text with ISO/IEC.)

3 ITU-T. *Information technology : Open distributed processing : Reference Model : Architecture.* Geneva, ITU, 1995. (ITU-T Rec. X.903.) (Common text with ISO/IEC.)

4 ITU-T. *Information technology : Open distributed processing : Reference Model : Architectural Semantics.* Geneva, ITU, 1997. (ITU-T Draft Rec. X.904.) (Common text with ISO/IEC.)

5 TINA-C. *Overall Concepts and Principles of TINA, Version 1.0.* 1995.

6 ITU-T. *Management of the transport network : Enterprise viewpoint for simple subnetwork connection management.* Geneva, ITU, 1996. (ITU-T Rec. G.852.1.)

7 ITUT-T. *Subnetwork connection management information viewpoint.* Geneva, ITU, 1996. (ITU-T Rec. G.853.2.)

8 IUT-T. *Management of the transport network : Computational interfaces for basic transport network model.* Geneva, ITU, 1996. (ITU-T Rec. G.854.1.)

9 ITU-T. *Management of the transport network : Application of the RM-ODP framework.* Geneva, ITU, 1996. (ITU-T Rec. G.851.1.)

10 ITU-T. *Principles for a Telecommunications management network.* Geneva, ITU, 1996. (ITU-T Rec. M.3010.)

11 ITU-T. *Generic functional architecture of transport networks.* Geneva, ITU, 1995. (ITU-T Rec. G.805.)

12 ITU-T. *Common elements of the information viewpoint for the management of a transport network.* Geneva, ITU, 1996. (ITU-T Rec. G.853.1.)

13 ITU-T. *Information technology : Open Systems Interconnection : Structure of Management Information : Guidelines for the definition of managed objects.* Geneva, ITU, 1992. (ITU.T Rec. X.722.) (Common text with ISO/IEC.)

14 ITU-T. *Information technology : Open Systems Interconnection : Structure of management information : General Relationship Model.* Geneva, ITU, 1995. (ITU-T Rec. X.725.) (Common text with ISO/IEC.)

15 Spivey, J M. *The Z Notation : A Reference Manual, 2nd ed.* Prentice Hall, 1992. (International Series in Computer Science.)

16 Kåråsen, A-G. The structure of OSI management information. *Telektronikk,* 89 (2/3), 90–96, 1993.

17 TINA-C. *Information Modeling Concepts, Version 2.0.* April 1995.

18 OMG. *The Common Object Request Broker : Architecture and Specification, Revision 2.2.* February 1998.

19 ITU-T. *Information technology : Open Systems Interconnection : Systems management overview.* Geneva, ITU, 1992. (ITU-T Rec. X.701.)

20 ITU-T. *Common management information protocol specification for CCITT applications.* Geneva, ITU, 1991. (ITU-T Rec. X.711.)

21 ITU-T. *Information technology : Abstract Syntax Notation One (ASN.1) : Specification of basic notation.* Geneva, ITU, 1994. (ITU-T Rec. X.680.) (Common text with ISO/IEC.)

22 ITU-T. *Common management information service definition for CCITT applications.* Geneva, ITU, 1991. (ITU-T Rec. X.710.)

23 ITU-T. *Information technology : Open Systems Interconnection : Structure of management information : Definition of management information.* Geneva, ITU, 1992. (ITU-T Rec. X.721.) (Common text with ISO/IEC.)

24 TINA-C. *Computational Modelling Concepts, Version 3.2.* May 1996.

25 TINA-C. *ODL Manual, Version 2.3.* July 1996.

26 Goldsack, S J, Kent, S J H (eds.). *Formal Methods and Object Technology.* Berlin, Springer, 1996.

27 Stepney, S, Barden, R, Cooper, D. *Object Orientation in Z.* Berlin, Springer, 1992.

Håkon Lønsethagen is Research Scientist at Telenor R&D, Kjeller, Network and Service Management Platform Unit. Currently, his focus is access network management and ATM transport network management. His research interests are systems architecture, systems evolution and specification techniques.

e-mail:
hakon.lonsethagen@fou.telenor.no

Engineering Communicating Systems

KNUT JOHANNESSEN

This paper presents a framework for configuration design of object-oriented communicating systems. System characteristics depend on the distribution of computing tasks between computing resources and many distribution schema are normally valid. Selection of the best overall distribution schema is a difficult optimization problem that has been studied in the context of file allocation and task allocation. Each distribution schema requires a specific support from the distributed processing environment. QoS attributes are used to guide the refinement of a computational viewpoint specification into an engineering viewpoint specification. Using workload information, hierarchical cluster analysis is proposed as a heuristic method to select and modify engineering clusters.

1 Introduction

The size and complexity of communicating systems span from a few computers interconnected on a local area network to global telecommunications networks. However, whatever the size and complexity, all communicating systems are interconnected by a network. Although the capacity of both wide area and local area networks is rapidly increasing, network interconnection usually results in different communication characteristics than communication within a single machine; lower bandwidth, longer message delay, other failure semantics, other security threats, etc. Design and configuration of communicating systems must take into account the network characteristics. Often, additional software components are required, e.g. in order to handle faults, to what would be necessary with a single computer. Also, the configuration of system components depends on the network characteristics.

In this paper, we use the term “engineering” to denote the assignment of computing tasks to computing resources, normally with the objective to achieve the overall best possible service. Communicating systems are not new, and there is a rich literature on engineering aspects such as task allocation (assignment) (e.g. [34, 2, 42, 7]) and load balancing (e.g. [29, 6]). The large volume of work related to engineering is an indication of the theoretical challenge, the practical requirements, as well as the diversity in considerations and system constraints.

In this paper, the focus is on object-oriented communicating systems and architectures. Some of the current research into architectures and technologies for future telecommunication systems, e.g. the Telecommunication Information Network Architecture (TINA) [5], is based on object-orientation. Management of telecommunication networks and services is a domain where object-oriented technology is increasing in importance. Many teleoperators have or have had systems dedicated to the management of a specific technology. This scenario is illustrated in Figure 1 where each technology (PDH, PSTN, etc.) is managed by one or more systems constituting a vertical slice through the logical management layers of the TMN architecture [23]. The result is (hopefully) a *technically* efficient solution within each domain. However, the overall architecture lacks horizontal integration, resulting in more difficult management across domains. Integration between systems is facilitated by manual operations, and the *organizational* efficiency suffers. In addition, some data elements are normally replicated between management domains, and the lack of integration makes it more difficult to maintain consistency. Hence, reduced data quality is a likely result of the vertically integrated architecture. Horizontal integration is required to achieve common management between technology domains. This scenario is shown in Figure 2 where some element management aspects are still handled by technology specific systems, while network and service management functions are handled by generic (technology independent) management systems. The result is better organizational efficiency with reduced need for manual operations. Horizontal integration also contributes to better data quality. However, the technical efficiency may be reduced with increased distance between components, which was (or would be) located together with vertical integration. A communication architecture based on the principle of a software bus for interaction between objects, e.g.

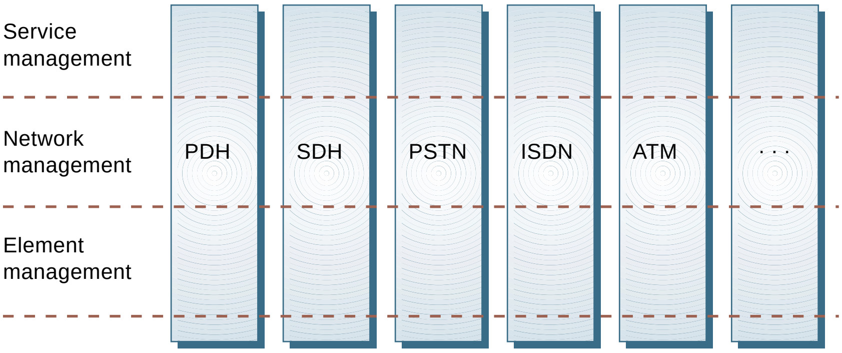


Figure 1 Vertical management system integration

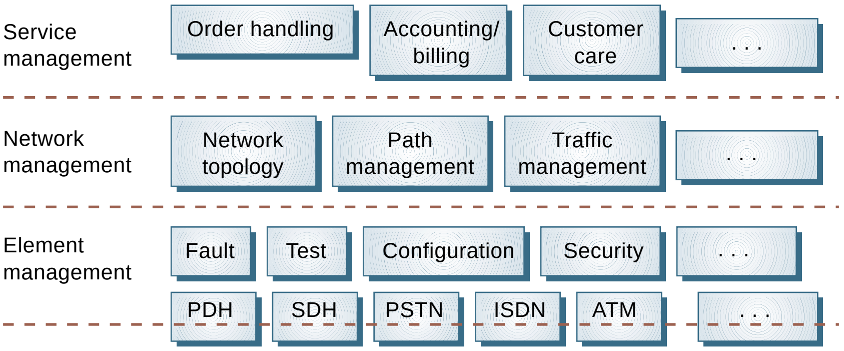


Figure 2 Horizontal management integration

CORBA [37], offers great flexibility in location of objects. By using technology based on this principle together with proper engineering, we may aim for both the functionality resulting from horizontal integration and the performance provided with vertical integration.

Components of communicating systems can fail during operation, and the load on the systems will change over time. In both cases, there is a need for reconfiguration. Efficient reconfiguration is only possible with support from the run-time environment. Communicating systems are able to continue operation with some reduction in performance and dependability[1] with one or more failures. *Performability* analysis (e.g. [36, 20]) is the analysis of the combined effect of performance and dependability on operational systems, and a performability manager [16] can use this analysis to maintain a required service level.

In principle, the capabilities that allow reconfiguration of systems in response to failures and change in load also make it possible to tune performance while the systems are in operation. Hence, it could naïvely be assumed that engineering is of minor importance during the development phase. Although communicating systems may allow reconfiguration during operation, this freedom is normally restricted by the components of the system. These components are identified and designed as part of the system development, when different system structures are analyzed to select the best set of components. The knowledge that is gained through this analysis is also of value when the system is put into operation. Further, more fundamentally, the engineering effort during development establishes confidence that the system – with specific configurations – will be able to meet service requirements.

In general, system specifications (or some parts of these) can be classified as being either functional or quantitative. The functional specification prescribes *“what the system shall do”* while the quantitative specification defines *“how well”*. In this paper, the functional specifications are taken as given, and concepts and techniques are presented that facilitate quantitative modelling during the development phase.

1.1 Outline

The reference model for open distributed processing (RM-ODP) is a useful framework for design of distributed systems and is presented in section 2.

The next subsection presents an overview of the allocation problem of object-oriented systems. The object model itself is sufficient for functional analysis (e.g. specification validation), but information about the computing infrastructure (nodes and network) is required for quantitative analysis. Hence, objects must be associated with concrete computational resources.

Interaction between objects must be supported by a run-time environment where the objects are assigned to different nodes. The required support from the infrastructure can be expressed as quality of service attributes as explained in section 3.

Performance engineering is a central part of a quantitative analysis, and techniques are available for use during system development. Important concepts from performance engineering are presented in section 4.

In section 5, the general concepts from performance engineering and analytical system analysis is applied to communicating systems. Workload modelling is presented together with system and model constraints, and possible solution techniques are identified.

1.2 The problem

Engineering of communicating systems is concerned with allocation of entities – in this case objects – to the computing resources that are available. The allocation should result in a configuration that meets the requirements to the services. Requirements at a high level can be stated in terms of functionality, quality and cost. Performance and dependability are aspects of quality and can be expressed as constraints (e.g. average interactive response time less than Δ seconds) associated with the functional requirements and capabilities. Often, we seek a system configuration (or architecture) that minimizes the total cost of the system. Engineering the distribution of communicating systems is clearly an optimization problem. In operational research this type of problem is generally known as the assignment problem [46]. The problem is visualized in Figure 3, where application objects map to nodes (the computing resources) which are interconnected by some communication network. An optimal solution to the allocation problem as formulated is not trivial. Even so, the above problem statement is a simplification. The problem is actually modified by the solution to the problem: interaction between application objects assigned to different nodes must be supported by distribution support objects not present in the original model. Distribution support objects perform basic communication functions (which are always required) and support the required quality of service associated with interactions. Distribution support objects are often allocated to the same node(s) as the supported application objects, but can also be assigned to separate nodes. This is shown in Figure 4.

2 Open Distributed Processing

The reference model for open distributed processing (RM-ODP) [25, 26] is both a framework for discourse and a basis for specification of concrete architectures for object-oriented distributed processing. This section identifies and describes the sub-set of these concepts that is important to engineering of communicating systems[2]. General introductions to ODP concepts are found in [40, 33].

2.1 Viewpoints

This section provides a brief discussion of viewpoints and their relevance to distribution engineering. A number of different abstractions of a system is possible. These abstractions are called viewpoints in RM-ODP and are formed by a selected set of architectural concepts. The reference model identifies five viewpoints as necessary and sufficient:

- *Enterprise viewpoint*
  The purpose and policies of the ODP system are issues in the enterprise viewpoint. In our context, the security policy is of particular interest.

[1] *The functionality of a system can be reduced when components fails, i.e. the availability of functions is reduced.*

[2] *RM-ODP use the term “system” in a general sense as “something of interest as a whole or as comprised of parts” [25].*

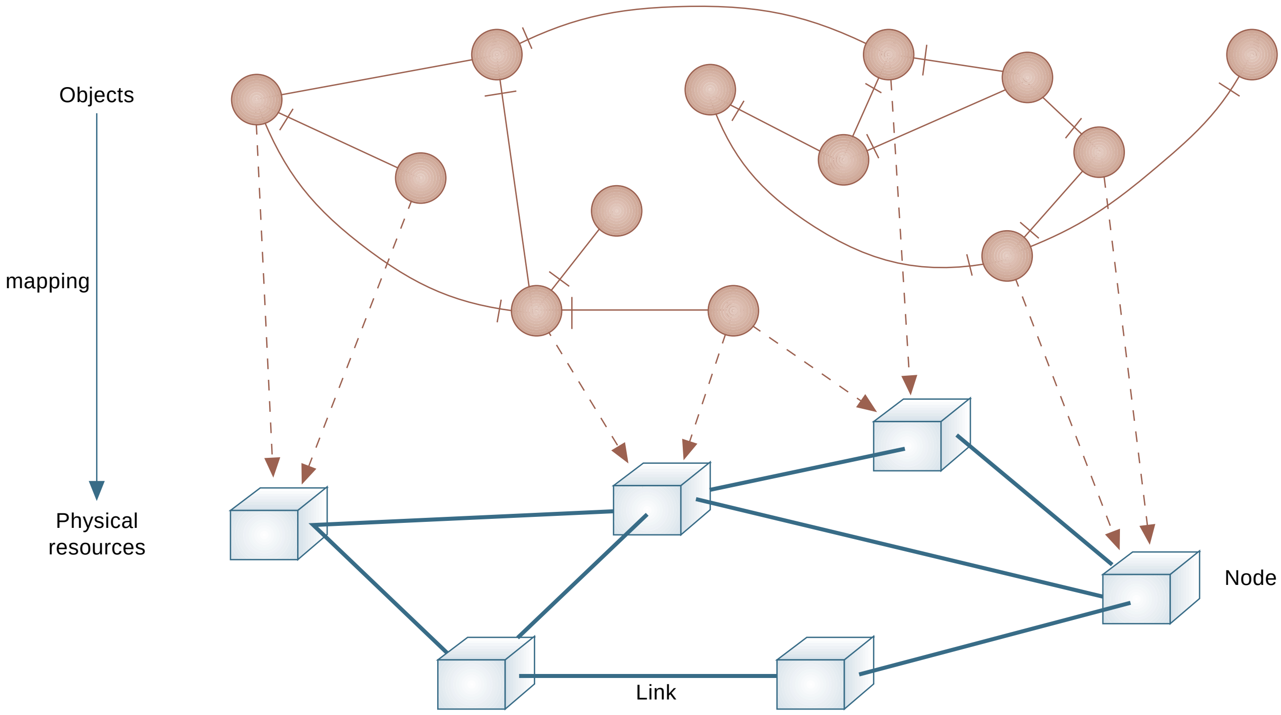


Figure 3 The assignment problem

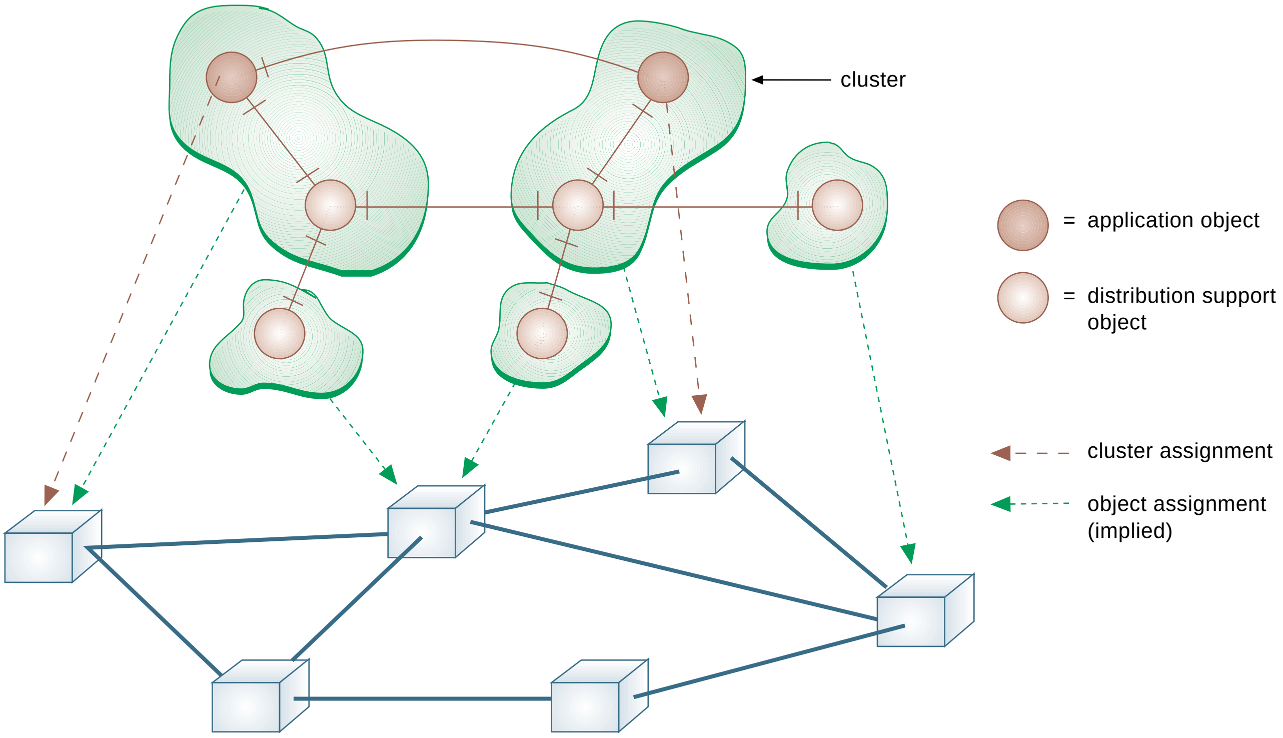


Figure 4 The effect of distribution on the assignment problem

- *Information viewpoint*
  The information viewpoint is concerned with the static and dynamic aspects of information in the ODP system and will not be addressed further.
- *Computational viewpoint*
  In the computational viewpoint, an ODP system is a collection of objects interacting at interfaces. Hence, the computational viewpoint enables distribution (e.g. partitioning, fragmentation and communication). Concepts from this viewpoint are discussed further in section 2.2.
- *Engineering viewpoint*
  The focus of the engineering viewpoint is on the mechanisms and functions in an ODP system that supports distribution. In a sense, the computational abstraction *prepares* for distribution (interaction between objects are exposed) while the engineering viewpoint provides a grouping of objects into larger units (clusters) that can be distributed within an ODP system. Concepts from this viewpoint are discussed further in section 2.3.
- *Technology viewpoint*
  This abstraction is close to implementation, and the technology that is selected for the implementation. The technology viewpoint will not be discussed further in this paper. However, a more complete engineering analysis than presented in this paper, will also take into account technology specific parameters.

The computational and engineering viewpoints are essential in our context, and key concepts from these viewpoints are described in the following two subsections.

2.2 Computational viewpoint

The computational viewpoint is an abstraction which focuses on objects interacting at interfaces. Interactions can be of three kinds: signals, operations and flows[3]. Signals are atomic interactions with unidirectional communication. The structure of a flow is important to both the producer and the consumer(s) of the flow. However, with respect to the distributed processing environment, a flow is an abstraction with no visible, internal structure.

Operations are either announcements or interrogations. Announcements are interactions (sent) from a client to a server, while interrogations consist of two interactions: first, an invocation by the client on the server and secondly, a termination returning information to the client.

In order for an interaction to take place between two (or more) objects, a binding must be established between the objects. A binding is either explicit, i.e. established through a binding action of a requesting object, or implicit, i.e. established when required by the infrastructure of the distributed system. A direct (primitive) binding is possible between two objects, while a binding between more than two objects is performed by a separate binding object.

2.3 Engineering viewpoint

The engineering viewpoint prescribes an implementation in terms of engineering objects, nodes, nucleus, capsules and cluster, and relations between these concepts. A node contains a nucleus (e.g. the operative system), a number of capsules and a capsule manager. Each capsule is a collection of objects forming a single unit for the purpose of encapsulation of processing and storage, e.g. an application process. Each capsule may contain a number of clusters together with a cluster manager. The cluster is a collection of engineering objects forming a distribution unit (e.g. distribution functions apply to clusters, not to single objects).

Bindings between engineering objects are either local or distributed. A channel is required for a distributed binding. The channel is a configuration of special engineering objects (stubs, binders, protocol objects and interceptor objects) between interfaces. A stub object is responsible for local transformations of the interactions in the channel, the binder objects maintain a distributed binding between the interacting objects, the protocol objects perform the communication protocol functions, and the interceptor object may perform required transformations as well as monitor and enforce policies on the permitted interactions. An example channel is shown in Figure 5.

In the engineering viewpoint specification, a particular allocation of objects to clusters is selected from the set of pos-

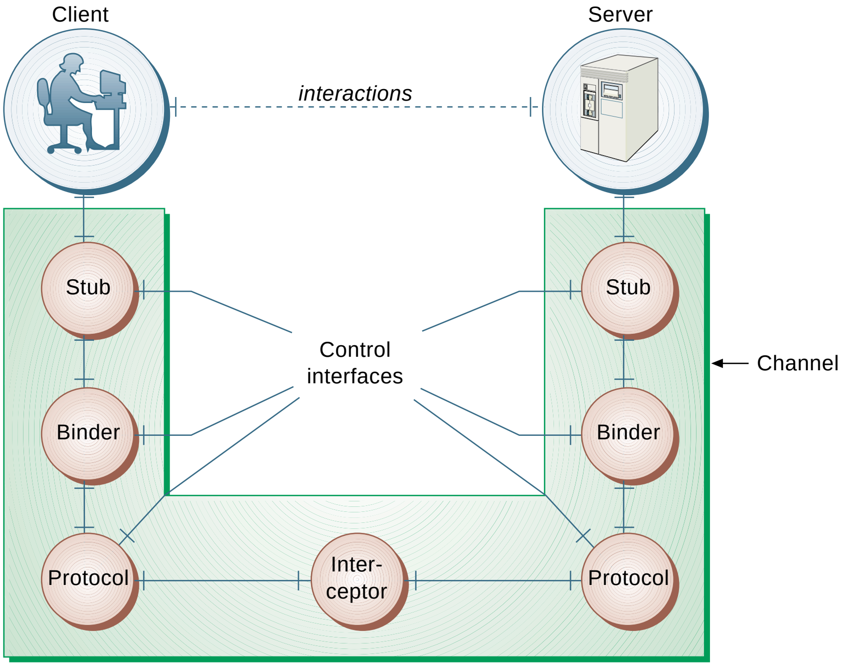


Figure 5 Channel in support of distributed binding

[3] *RM-ODP distinguishes between interactions that may not be visible and interfaces that are visible. This distinction is reflected in the terminology. Hence, a "flow" is an interaction that can be made visible across a "stream" interface.*

sible allocations allowed by the computational viewpoint specification. In this paper, the allocation of object to clusters is called the distribution schema. It should be noted that the distribution schema does not (necessarily) include assignment of clusters to capsules (processes) on particular nodes. However, derivation of the distribution schemata is based on a (possibly idealized) specification of available computing resources.

2.4 Distributed processing environment and distribution services

An attractive feature of ODP systems is the (promised) set of capabilities offered by the distributed processing environment (DPE). These capabilities are available to construction of ODP systems either explicitly, as services (ODP functions), or implicitly through distribution transparencies.

Distribution transparencies allow the developer to focus on the design and build of his local functions without having to know the implementation or location of co-operating functions. In effect, a distribution transparency hides supporting services as well as implementation issues from the developer. The result is reduced complexity as seen from an application perspective, but at the cost of increased complexity of the infrastructure [43]. The reference model (RM-ODP) identifies the following transparencies:

- Access transparency masks differences in data representation and invocation mechanisms (i.e. protocols).
- Failure transparency masks from an object the failure of other objects (or the object itself) to enable fault tolerance.
- Location transparency masks the use of location information from binding actions.
- Migration transparency masks from an object the systems' capability of moving that object.
- Relocation transparency masks relocation of an interface from other interfaces bound to that interface.
- Replication transparency masks the use of replicated objects in support of an interface.
- Persistence transparency masks from an object the activation and deactivation of other objects (or the object itself).
- Transaction transparency masks co-ordination between objects to achieve consistency.

In contrast to distribution transparencies which are hidden from the application, ODP functions (also known as DPE services) are accessed directly from the application code, and are a collection of functions that are applicable to the construction of distributed processes [26]. The reference model identifies four groups of ODP functions:

- *Management functions*
  These functions are related to management of nodes, objects, clusters and capsules.
- *Co-ordination functions*
  This group includes functions for event notification, checkpoint and recovery, object grouping, replication, object migration and transaction processing. Note: While migration transparency masks from the object that it is being moved, the DPE provides capabilities allowing *other* objects to control object migration.
- *Repository functions*
  Storage functions, relocation function and trading functions are part of this group. A trader is a support service matching requests for service with objects offering a compatible service [27].
- *Security functions*
  The reference model identifies a set of security services equivalent to the security services defined in the security framework [5], i.e. authentication and access control, integrity and confidentiality, non-repudiation as well as security audit and key management.

The capabilities of the DPE reduces the burden on the developer. However, in a deployed ODP system, distribution services are activated and will influence the performance of the system. The effect of this influence is not obvious and is an additional challenge to performance engineering.

3 Refinement of specifications

During development, a system is defined at several levels of abstraction from an initial high level specification through design to detailed implementation. Each level of abstraction makes some characteristics of the system visible, while other characteristics are disregarded. We use the term "non-functional" to denote requirements or properties associated with the functional aspects that are defined at a lower level of abstraction. These properties are used (together with other information) to guide the refinement or transformation into a more detailed specification. Location of objects and clusters (the computing tasks) is seen as a refinement of the computational specification into an engineering specification.

The notion of "non-functional" requirements is similar to QoS as used in [22], which identifies QoS requirements on objects (e.g. capacity), on interactions (e.g. timeliness and security) and associated with content (e.g. integrity). However, on operations only a subset of the QoS characteristics identified in [22] is relevant (e.g. stream oriented requirements such as jitter may not be relevant).

In general, QoS requirement can be stated as attributes from three perspectives:

- *Offered*
  These are QoS attributes associated with an interface supported by a server object and thus available for inspection and selection by a client.
- *Required*
  These are QoS requirements expressed by client objects and which must be met by server objects (and the distributed processing environment) to have a successful binding.
- *Expected*
  A server object may express QoS requirements to be met by the client object. Such requirements are less intuitive than *offered* and *required* QoS, but are equally relevant. One example is traffic shaping on broadband stream interfaces. Another example is a banking application where the bank *expects* the client (customer) to support security services (e.g. authentication and integrity) on interactions.

In addition, the *agreed* QoS is the negotiated QoS between the client and the server based on the required, offered and expected QoS requirements.

The following subsections describe how security and dependability requirements are expressed as QoS attributes on interfaces and how model refinement is guided by a set of transformation rules.

3.1 Security requirements

Security requirements are often expressed in terms of confidentiality, integrity and availability and are supported by security services. A general framework of security services is given in [24], which identifies the following five service classes:

- Authentication
- Access control
- Confidentiality
- Integrity
- Non-repudiation.

Each class may contain more specialized services, and [24] identifies a total of 14 security services. Security services are abstractions and do not (as the name could indicate) identify any particular type of implementation. In fact, a security service can sometimes be realized by different security *mechanisms*, depending on the context. Encipherment is an example of a mechanism which can support a number of services. Some mechanisms are pervasive, i.e. not specific to a subset of security services, and are in principle useful to all services. Trusted functionality and audit trail are examples of such mechanisms.

Engineering of the security requirements can be organized into three steps:

1. First, security requirements are stated using QoS attributes which identify security services.
2. Second, security mechanisms are selected to suit the distribution between client and server objects, e.g. confidentiality using access control is possible within a single cluster or node, while confidentiality must be supported by encipherment when information is communicated across an exposed network.
3. As the final step, a specific implementation of the security mechanism is selected, e.g. a large number of encipherment algorithms are (or may be) available.

3.2 Dependability requirements

Dependability [31] is expressed by a number of attributes; usually related to

- reliability
- availability
- safety[4].

Dependability objectives are expressed in terms of the three dependability attributes, and the systems are designed to meet these objectives through the use of dependability techniques. Dependability is designed by observing

- fault tolerance
- fault prevention
- fault removal
- fault forecasting.

Of these techniques, fault tolerance is the capability of a system to continue operation (possibly with reduced performance) in the presence of faults; a fault is masked and not seen as a failure from the environment (e.g. the user). The aim of the other techniques is to remove faults before the system is put into operation. As we are concerned with system characteristics resulting from the distribution of objects and clusters, the techniques for fault tolerance handling is of most relevance.

While security attributes state the permissible capabilities offered by an object, the corresponding dependability attributes state the ways in which the object is able to fail[5]. A failure is manifested in a failure behaviour (the failure semantics) and the fault tolerant capabilities must be constructed accordingly. Failures can be classified as *timing* or *value* failures [8]. Timing failures can further be identified as *early, late* or *omission* timing failures.

The failure semantic of an object is expressed as a subset of the set of failure classifications:

```
Dependability ::= SEQUENCE {

  failureMode
  FailureModes
  OPTIONAL,

  -- failureMode is only
  -- present as part of the
  -- offered QoS expressed
  -- by a server object

  availability
  DependabilityMeasure
  OPTIONAL,

  reliability
  DependabilityMeasure
  OPTIONAL

}

FailureModes ::= BIT STRING {
  unknown (0)
  valueFailure (1),
  timingFailure (2),
  earlyTimingFailure (3),
  lateTimingFailure (4),
  omissionTimingFailure(5),
  ...
}

DependabilityMeasure ::= REAL
```

Security and dependability requirements and characteristics can be combined into a single QoS attribute.

3.3 Transformation rules

In principle, each object in an ODP system can be assigned to a cluster independently of all other objects. The number of possibilities for assignment of *n* objects to clusters is given by the number of ways to assign *n* items to subsets. With an unlimited[6] number of clusters, the number of possible distributions is given by the Bell number [18]:

$$B(n+1) = \sum_k \binom{n}{k} B(n-k)$$

$B(n) \approx n^n$, for large n

With the number of clusters fixed to *K*, the number of distributions is given by [28]:

[4] *Some authors [e.g. 8] also treat information security as an aspect of dependability. However, the relationship between dependability and security is not fully clarified, and for the present discussion they are therefore treated as different properties. Safety is not considered in this paper.*

[5] *The failure modes of containing cluster, capsule and node do of course influence the possible failure modes of an object. Hence, the specification of failure modes on objects will be considered as requirements to be supported by the DPE.*

[6] *The number of objects is an upper limit on the number of clusters.*

$$S(n,K) = S(n-1,K-1) + K \cdot S(n-1,K)$$

$$S(n,K) = \frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}i^{n}$$

Example: With $n = 50$ the number of distributions is roughly 10^{47}. If the number of clusters is fixed with $K = 5$, the number "reduces" to 10^{32}.

In practical systems, restrictions are likely to reduce the degree of freedom in objects assignment. However, it is impossible to manually design an optimal[7] distribution schema of many interesting systems. Rules are required to guide the automated transformation of a computational viewpoint specification into an engineering viewpoint specification.

Transformation rules must capture both the characteristics of the assignment process, as well as relationships between QoS attributes.

Transformation rules are invoked when object instances are allocated to different clusters. When this happens, the interactions (if any) between the objects are examined to determine if QoS requirements are defined that requires a refinement of the specification (see Figure 6).

A layered approach is assumed in the transformation, i.e. the different QoS requirements are treated as separate transformations. Another approach is an integrated transformation where all QoS requirements (on one interface) are considered at the same time. There are arguments in favour of and against both the layered and the integrated approach [14]. The sequence in which the transformations are applied is significant, and the resulting protocol hierarchies are shown in Figure 7.

We assume that all transformation follows the second alternative to ensure that all communication is protected as appropriate.

An outline of the security transformation function is provided in Figure 8. The result of the transformation is a sequence of specification transformation steps. Since a transformation may not result in an optimal distribution (or better than the

DISTRIBUTION TRANSFORMATION

Problem: Client object *c* and server object *s* are to be assigned to different clusters.

Answer: Δ, i.e. the list of specification transformation steps to meet the QoS requirements.

Comment: Interaction, e.g. interfaces between object *c* and object *s* are examined to determine the need for specification transformation.

```
begin
    $\Delta \leftarrow \emptyset$; (* an empty set of transformations *)
    if assignment(c) $\neq$ assignment(s) then do
        forall interface $\in$ interaction(c,s) $\cup$ interaction(s,c) do
            begin
                if security QoS defined on interface then
                    $\Delta \leftarrow \Delta$ + transformation sequence of security support;
                if dependability QoS defined on interface then
                    $\Delta \leftarrow \Delta$ + transformation sequence of dependability support;
            end
    return $\Delta$;
end
```

Figure 6 Distribution transformation

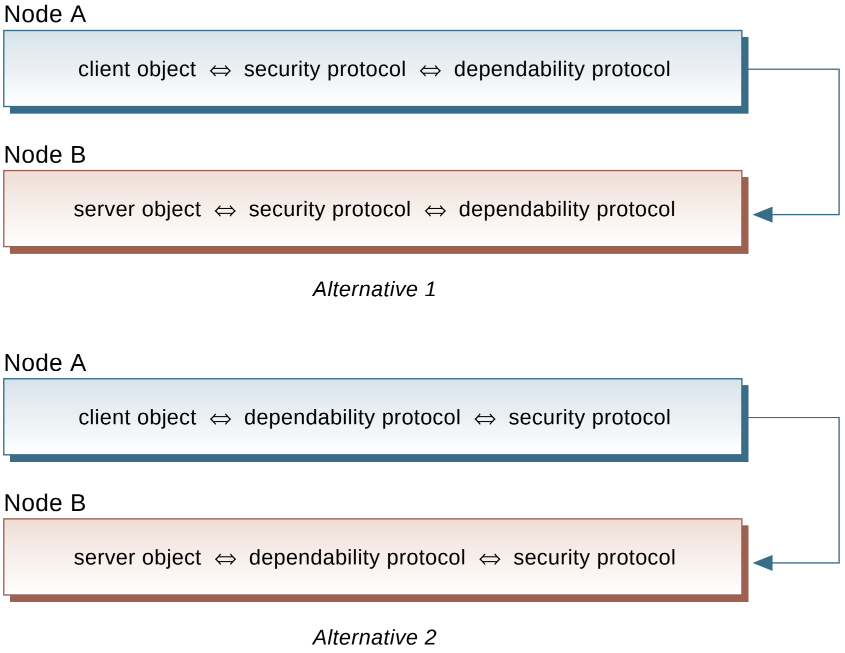

Node A
client object ⇔ security protocol ⇔ dependability protocol
Node B
server object ⇔ security protocol ⇔ dependability protocol
Alternative 1
Node A
client object ⇔ dependability protocol ⇔ security protocol
Node B
server object ⇔ dependability protocol ⇔ security protocol
Alternative 2

Figure 7 Security and dependability protocol hierarchies

[7] *With respect to some stated objective.*

current), backtracking is essential and this task is made easier with a transformation sequence.

3.4 Transformation example

An example computational specification with three objects A, B and C is shown in Figure 9 where arrows point towards the server object in each interaction. This example also illustrates that the presence of security requirements can guide the identification of clusters. In this case, object A expects a non-repudiation service on the interface used by object B. Normally, non-repudiation services are only considered on interactions between objects in different domains. Hence, objects A and B are most likely assigned to different clusters.

A concrete interpretation of this example is the following:

Object A represents an ordering service. Non-repudiation with proof of origin is used to ensure the authenticity of the purchaser (object B) and the content of the order. Object B makes use of a subscribed information service (object C) and must present access control information to get access. Another information service from object C is used by object A.

This simple specification is transformed into the engineering viewpoint specification shown in Figure 10. In this case, the offered and required dependability attributes are compatible, and the dependability transformations are not invoked. Authentication is supported by authentication service objects (ASO), which generate and validate authentication information. Access control information is added to the interaction between B and C by the AIO object; this information is received by the access enforcement object (AEO) which queries the access decision object (ADO) to determine if access is permitted.

The non-repudiation requirement introduces a little more complication. The non-repudiation service requires the support of an integrity service. Hence, between the object generating non-repudiation information (NGO) and the user of this information (NUO), integrity service objects (ISO) are inserted to ensure the integrity of non-repudiation information. In this case, the non-repudiation is further supported by a trusted third party (TTP).

SECURITY TRANSFORM

Problem: Client object c and server object s with security requirements shall be allocated to different clusters.

Answer: A sequence of engineering specification transformations (Δ) is provided to meet the security QoS requirements.

Comment: Security requirements are validated and adjusted to add basic security services required by high-level security services, e.g. non-repudiation implies authentication and integrity, and to remove redundant basic services, e.g. integrity is made redundant by confidentiality.

Security capabilities are added to the model in the following order:

- access control
- non-repudiation
- authentication
- confidentiality
- integrity

This sequence is a design decision imbedded in the transformation rules.

```
begin
    required_c $\leftarrow$ {required security capabilities from c};
    expected_s $\leftarrow$ {expected security capabilities from s};
    $\Delta \leftarrow \emptyset$; (* an empty set of transformations *)

    if not compatible requirements and capabilities then
        error "object c and s cannot be located in different clusters with fulfilment of security requirements";

    sec_req $\leftarrow$ required_c $\cup$ expected_s;
    seq_req $\leftarrow$ seq_req $\cup$ {basic services required by high level services};

    for capability $\in$ {access control, non-repudiation, authentication, confidentiality, integrity} do
        if capability $\in$ seq_req then
            begin
                $\Delta \leftarrow \Delta$ + new security service objects if not already present;
                $\Delta \leftarrow \Delta$ + interfaces within the clusters and between the clusters;
                $\Delta \leftarrow \Delta$ + remove old interface between the clusters when inline security service;
                $\Delta \leftarrow \Delta$ + attach/update workload information to interfaces:
            end
    return $\Delta$;
end
```

Figure 8 Security transformation

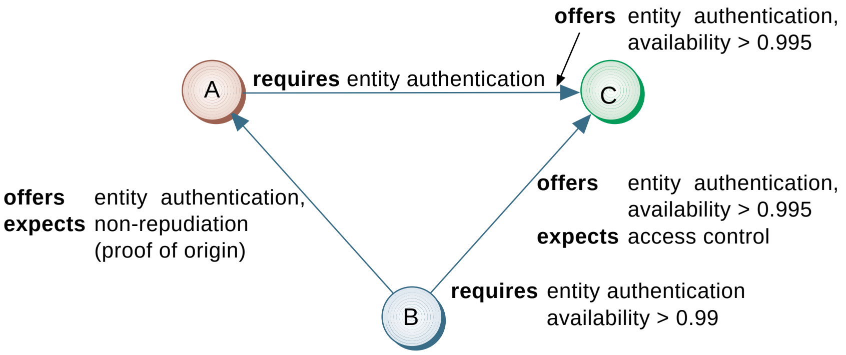

offers entity authentication, availability > 0.995
A
requires entity authentication
C
offers entity authentication,
expects non-repudiation (proof of origin)
offers entity authentication, availability > 0.995
expects access control
B
requires entity authentication availability > 0.99

Figure 9 Application model with security requirements on interactions

Of course, the transformation shown in Figure 10 is only valid – or of interest – when the three objects A, B and C are assigned to different clusters. Another allocation could give a different transformation.

4 Performance engineering

The goal of performance engineering is to establish quantitative knowledge of the performance of a computer system. This knowledge is expressed using a set of performance measures for the different characteristics of the system. In this paper, we are concerned with performance (real or predicted) of deployed systems only. Other important performance issues, e.g. related to systems development, are not addressed. Performance measures are always related to a measurement context (measurement period, time and day of measurement, etc.) which should be stated together with the relevant performance measure. A classification of system performance measures is given in Table 1.

Within the literature, both *performance evaluation* and *performance engineering* are encountered [15, 44]. There is no clear distinction between these terms. Historically, *evaluation* has roughly been associated with evaluation of existing, completed or nearly completed systems. However, it is well known that it is harder to remedy deficiencies in a finished product – or late in the development process – than to remove the problems in the design, and performance *engineering* stresses the importance of addressing performance in the design process [38].

The formulation of performance evaluation is simple: a workload is (thought to be) applied to the system, resulting in a performance prediction as shown in Figure 11. Workload is the totality of processing requests submitted to a system from the users during some period of time. Information system users are normally human, but in general the users can be both human and machine. Of course, a real workload can only be applied to a real system, and during development a model workload must be applied to a model of the system.

Performance (P) is a function of both the system structure (S) and the workload (W):

$P(S, W)$

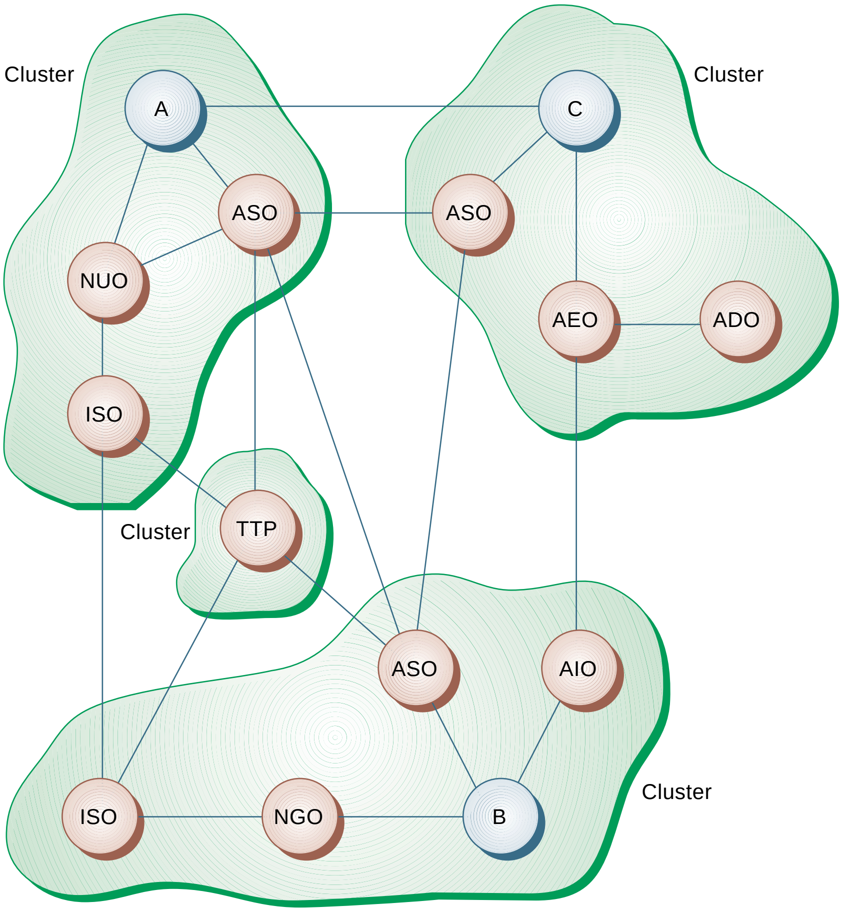


Figure 10 Example application augmented with distribution support objects

Table 1 System performance measures (adapted from [15])

Class	Definition	Example performance measure
Productivity	Volume of work per unit time (item/time unit)	Throughput rate Production rate Capacity (maximum throughput rate)
Responsiveness	The time between input and corresponding output (time unit)	Response time, e.g. transaction execution time Turnaround time, e.g. time to start a new job Reaction time, e.g. time between a key is pressed and the echo is presented on the screen
Utilization	The fraction of an interval in which a component is used (for some purpose) (dimension-less)	Hardware module utilization (CPU, I/O, ...) Operation system utilization Application utilization Database utilization

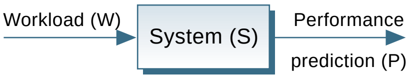


Figure 11 Performance evaluation

Notation

T	observation period
C_k	number of observed completions
X_k	throughput
B_k	busy period
U_k	utilization
N_k	number of customers
M_k	number of interactive users
R_k	residence time
Z	think time (terminal user)
V_k	visitation ratio

Note – A subscript denotes a specific service center or device

Operational relationships

throughput	$X \equiv \dfrac{\text{number of completions}}{\text{observation period}} \equiv \dfrac{C}{T}$
mean service time	$S \equiv \dfrac{\text{busy period}}{\text{number of completions}} \equiv \dfrac{B}{C}$
utilization	$U \equiv \dfrac{\text{busy period}}{\text{observation period}} \equiv \dfrac{B}{T}$

Operational laws

Little's law	$N = X \cdot R$
General response time law	$R = \sum_k R_k \cdot V_k$
Interactive response time law	$R = \dfrac{N}{X} - Z$
Forced flow law	$X_k = V_k \cdot X$

Figure 12 Operational analysis

Three different types of performance engineering activities can be identified [21]:

- *design*
  The goal of this activity is construction of a system structure (S) which provides a requested performance (p_0) when subjected to a specified workload, i.e. $P(S, W) \geq p_0$.
- *improvement*
  The performance of a system (S) can be improved if it is possible to modify the system to a new structure S' with $P(S', W) > P(S, W)$.
- *comparison*
  When n system structures are possible, performance prediction aims to provide an ordering between the systems:

  $P(S_1, W) \geq P(S_2, W) \geq ... \geq P(S_n, W)$

Several techniques are available for performance engineering. While exact measurements can be obtained on an existing system, performance predication must rely on a model of the proposed system. Mathematical models are often considered 'best', as analytical solutions are sometimes obtainable. However, not all mathematical models have solutions, or unrealistic assumptions must be made in order to produce a solution. In these cases, simulation models are attractive. Simulation is flexible, but at the cost of processing and interpretation of results. Furthermore, a simulation model (only) gives results to some level of confidence.

Although important, performance engineering is but one aspect of systems engineering. The methods used must be evaluated against a set of requirements [44, 38]:

- supplement (and integrate with) existing methods
- ease of use
- rapid assessment
- sufficient precision
- lifecycle usefulness
- wide applicability
- cost-efficiency.

Based on the above requirements, a significant part of work in performance engineering applies operational analysis [38, 10, 32] or mean value analysis [19, 44, 38, 15, 35, 32] rather than more elaborate (and complex) network queuing analysis. Operational analysis is based on relationships between observable variables over a period of operation (when the system exists). The power of operational analysis is founded in that these relationships are independent of distribution functions (as is Little's result). The utilization law serves as an example on how operational relations are established (notation is given in Figure 12), and is derived as

$$U = \frac{B}{T} = \frac{B}{T} \cdot \frac{C}{C} = \frac{C}{T} \cdot \frac{B}{C} = X \cdot S$$

Other operational laws are derived in a similar manner and can be applied to each service centre (both technical devices such as CPU and disk, and human processing tasks). The original formulations of the operational laws [3] was only applicable to open queuing networks, but have later been extended to cover also closed queuing networks [9].

An example illustrates the application of operational analysis:

Assume that a system has been observed over some time. During this observation period, the system has been used (busy) 75 % of the time and has processed (completed) 500 jobs (e.g. customers), and the mean service time per job is 50 ms. At the end of the observation period, 45 jobs are either being processed or are waiting. What is the response time of the system?

In this example, the throughput is

$$X = \frac{U}{S} = \frac{0.75}{50 \ \text{ms/job}} = 15 \ \text{jobs/s}$$

and the response time of the system is found as

$$R = \frac{N}{X} = \frac{45 \ \text{jobs}}{15 \ \text{jobs/s}} = 3 \ \text{s}$$

Mean value analysis (MVA) [38, 19] is a set of computationally efficient solutions to both open and closed queuing networks. Different service disciplines, e.g. infinite service, load dependent service, load-independent service, can be used to construct and solve a number of system models using MVA.

Operational variables can be considered a sample of the corresponding stochastic variable. A drawback with both operational analysis and MVA is that only expectation values are computed. Higher order moments (e.g. variance) or distribution functions of response time, utilization, etc. are not available. This kind of information is normally of significant value when the result of analysis is interpreted. An example is illustrative: With an exponential response time distribution, 22.3 % of all customers experience a response time in excess of 50 % of the mean value, and 13.5 % experience a response time twice as long as the average. If the response time had followed an Erlang-10 distribution (i.e. the distribution of the sum of ten identical, independent exponentially distributed variables), only 7.0 % of all customers would experience a response time exceeding 50 % of average and only 5.0 % would experience a response time exceeding twice the average. These concerns are to some extent addressed with sensitivity analysis [38] which can be used to evaluate the influence on a given performance measure from a change in either a performance parameter or a system parameter.

Operational analysis and MVA are favoured by performance engineers who argue [e.g. 44, 38] that these techniques fulfil the set of requirements presented above (ease of use, etc.). More detailed analysis, e.g. use of queuing theory, is sometimes based on assumptions that do not hold for computer systems (e.g. arrival and service time distributions).

Another reason why simpler mean value analysis is used, is the inherent inaccuracy in the workload parameters (presented in the following subsection), which reduces the utility of detailed analysis.

4.1 Workload

Modelling of the workload is one of the most difficult tasks of performance engineering. The workload is the combination of service requests put on the system from the environment. Performance engineers often identify three types of service based on the parameters characterizing the service usage [32]:

- *transaction*
  A transaction workload is the result of “infrequent” service request from a large number of users such that the workload is independent of the number of active users. Hence, the workload can be expressed by a single arrival intensity λ.

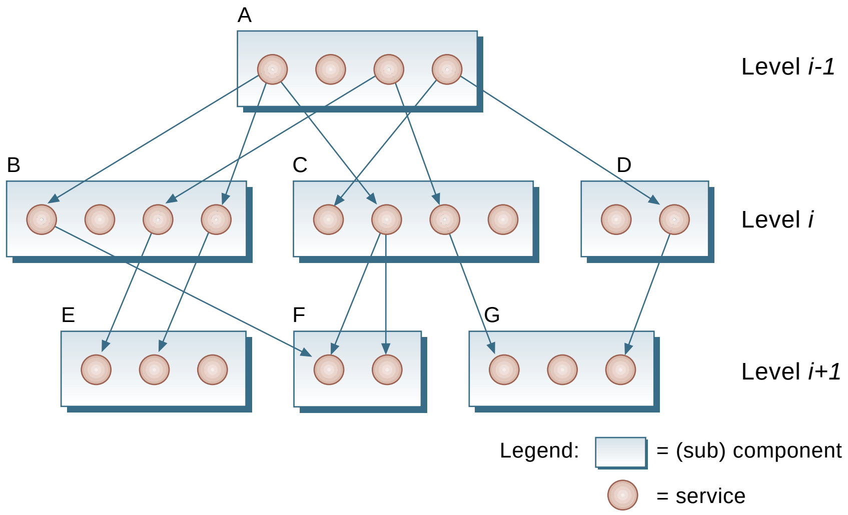


Figure 13 Generalized system structure (example)

- *batch*
  During the observation period, the batch workload is considered to be running. Hence, there is no arrival of new batch jobs and the workload is expressed by the size n_j of batch job *j*.
- *interactive* (or terminal)
  A *transaction* workload results when the infrequent requests are applied from a large number of users. If the number of users is smaller or the service request is more frequent, the resulting workload depends on the state, i.e. the number of active jobs. Hence, the use of an interactive service *k* is characterized by the average number of users of the service together with the per user intensity, i.e. (n_k, λ_k).

The total workload on a system can be identified as a combination of transaction, batch and interactive workloads.

An external service request, e.g. from a user, can be recursively decomposed to a sequence of requests[8] on subcomponents. This decomposition process is by some authors called *workload devolution* [21]. The devolved work describes service invocations between components and is independent of the rate of work, and is sometimes called a static model of the workload. The static model together with the dynamic invocation of services (e.g. transaction intensity), defines the dynamic workload on system components. The static model can be derived from inspection of the system and software structure. Figure 13 illustrates how invocation of services at some level *i* results in invocation of services from components at the next level. Service invocation between two adjacent levels of abstraction can be described as an $n_i \times n_{i+1}$ matrix

$$\underset{\sim}{C}_i = (c_{kl})$$

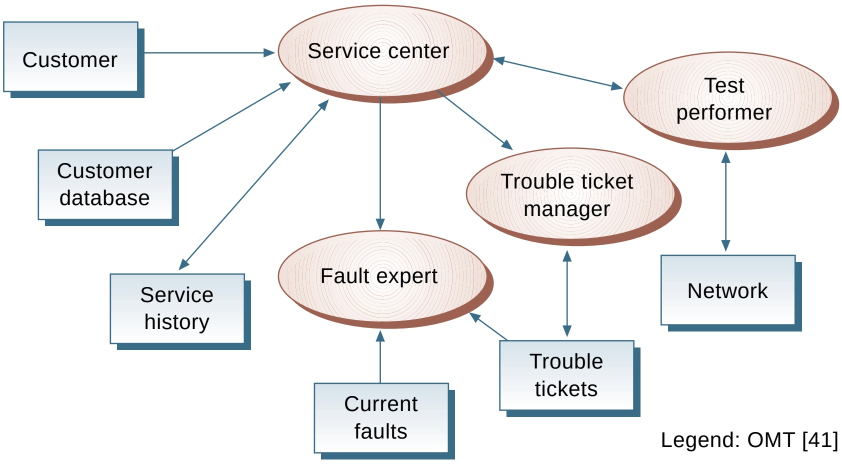
where n_i is the number of components at level *i*. From this specification it is easy to compute the invocation of the low-level *n* services as

$$\underset{\sim}{C} = \underset{\sim}{C}_o \times \underset{\sim}{C}_1 \times K \times \underset{\sim}{C}_n$$

Of course, many execution graphs are not strictly hierarchical as presented in Figure 13. Cycles within the graph are possible as a result of direct or indirect recursion. In such cases, the elements of $\underset{\sim}{C}_i$ must be replaced by the average number of invocations with the loops removed.[9]

[8] *It is useful to separate between service requests on an abstract machine and operations on a specific physical configuration. This separation allows for mapping to different implementations from the same basic model.*

[9] *This is always possible in a real (and stable) system where the number of recursive invocations must be finite.*



Legend: OMT [41]

Figure 14 Trouble management scenario

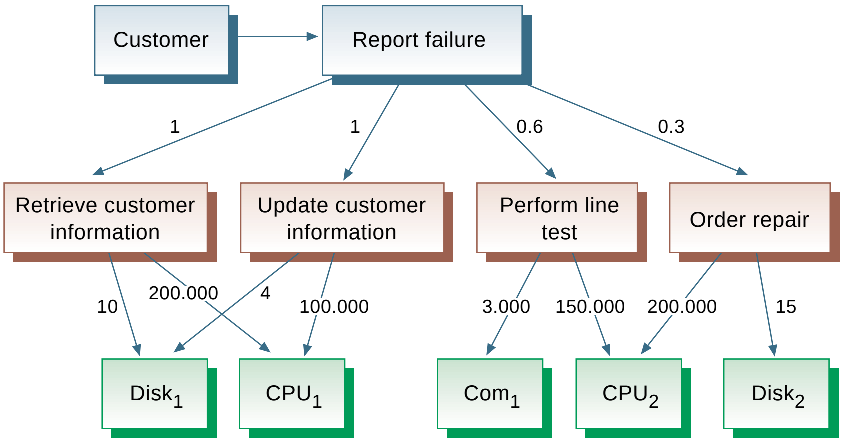


Figure 15 Trouble management configuration and workload (example)

4.2 Example

A simplified example from telecommunication trouble management illustrates workload modelling: Customers contact a service centre to notify of some fault in the network (e.g. missing dial tone). At the service centre, the following operations can be performed (also shown in Figure 14):

1 Information is retrieved on

- customer
- current network faults possibly affecting the customer
- the customer service history

2 Testing of access equipment

3 Create trouble ticket

4 Update customer service history.

The workload can be decomposed into database requests, CPU requirements and communications requirements (the low-level services). An example decomposition of one service request is shown in Figure 14: The report failure task is invoked by the customer. The customer information is always retrieved and is always updated, while line test is only performed for 60 % of reported failures and repair is only requested for 30 % of all reported failures.[10] Total workload on physical resources is then calculated as:

$$\underset{\sim}{W} =$$

$$\begin{bmatrix} 1 & 1 & 0.6 & 0.3 \end{bmatrix} \begin{bmatrix} 200000 & 0 & 10 & 0 & 0 \\ 100000 & 0 & 4 & 0 & 0 \\ 0 & 150000 & 0 & 0 & 3000 \\ 0 & 200000 & 0 & 15 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} 300000 & 1500000 & 14 & 4.5 & 1800 \end{bmatrix}$$

where the resulting vector is workload on (CPU$_1$, CPU$_2$, Disk$_1$, Disk$_2$, Com$_1$), and stated in terms of CPU cycles, disk access and message length.

5 Distribution engineering

The objective of distribution engineering of systems is to identify an optimal assignment of objects to computing resources. The next subsection explains workload modelling of interacting objects. Subsequently, constraints on the optimization problem[11] are identified. Different optimization objectives are presented in section 5.3 and an overview of possible solution techniques are given in 5.4.

5.1 Workload modelling

The workload model contains a static part, i.e. how objects interact, as well as a dynamic part, i.e. the frequency of interaction as described in section 4.1. It is useful to classify objects as either *core objects* or *user objects* such that there is no interaction between user objects. The purpose of this classification is to separate the workload generating part, e.g. the user objects. The total workload is then the sum of work generated by all user objects.

The static structure of the workload is derived from object behaviour information. For each run (activation) of a user object *u* we record the number of invocations of operation *o* on core object *s* by object *c* as $s^{u}_{(c,s,o)}$. Restricting the workload to transactions, the interaction frequency between objects is then computed as

[10] *All figures in the example are for illustrative purposes only and are not observed from operational trouble management.*

[11] *An overview of optimization models can be found in [46].*

$$\lambda^u_{(c,s,o)} = \lambda_u \cdot s^u_{(c,s,o)}$$
$$\lambda_{(c,s,o)} = \sum_u \lambda^u_{(c,s,o)}$$
$$\lambda_{(s,o)} = \sum_c \lambda_{(c,s,o)}$$
$$\lambda_{(s)} = \sum_o \lambda_{(s,o)}$$

where $\lambda_{(s)}$ is the aggregated activation frequency on core object *s*.

Given a processing requirement P^C_o per operation *o* of object class *C*, the resulting per time unit processing requirement for core object *s* is computed as

$$p_{(s,o)} = \lambda_{(s,o)} \cdot P^{C(s)}_o$$
$$p_{(s)} = \sum_o p_{(s,o)}$$

The user object rate λ_u and the processing requirement P^C_o are difficult to estimate and the uncertainty limits the precision of the workload model. The invocation structure $s^u_{(c,s,o)}$ can be derived from simulation of the behaviour of the system or directly counted from trace information. Of course, exhaustive counting from all possible trace sequences is not realistic in a real system. However, the most frequent execution paths can be expected to dominate the result. Early in the design phase, detailed information of the processing requirement P^C_o is not available, and educated estimates must suffice. Hence, the workload model needs to be revised in an iterative process as more information is made available.

5.2 Constraints

Capacity constraints are related to processing, storage and communication:

- The requested processing from all objects and clusters assigned to a node must be less than the available capacity $C^P_{(n)}$ of the node:

  $$\sum_{c \in A_C(n)} \sum_{s \in A_o(c)} P(s) < C^P_n$$

  for all nodes *n*

  where $A_C(n)$ is the set of clusters assigned to node *n* and $A_O(c)$ is the set of objects in cluster *c*.

- Total bandwidth requirements from object interactions must be less than the bandwidth capacity C^L_l available on links between nodes:

  $$\sum_{(s,c,o) \in A_L(l)} m_o \cdot \lambda_{(c,s,o)} < C^L_l$$

  for all links *l*

  where m_o is the mean message size for operation *o* and $A_L(l)$ is the set of all interactions routed on link *l*.

- Physical storage capacity C^S_n of each node must be greater than the required storage from all objects.

  $$\sum_{s \in A_o(n)} d_{(s)} < C^S_n$$

  for all nodes *n*.

In addition to the capacity constraints, the model formulation itself has a set of constraints, e.g.

- an object is allocated to only one cluster
- all objects are assigned to some cluster
- a cluster is allocated to only one computing node
- all clusters are assigned to some node.

Given the following definitions

$$\Theta_{i,j} = \begin{cases} 1, \text{ if object } i \text{ is allocated to cluster } j \\ 0, \text{ otherwise} \end{cases}$$

$$\Psi_{i,j} = \begin{cases} 1, \text{ if cluster } i \text{ is allocated to node } j \\ 0, \text{ otherwise} \end{cases}$$

$$\Omega_i = \begin{cases} 1, \text{ if any object is allocated to cluster } i \\ 0, \text{ otherwise} \end{cases}$$

$$\Phi_i = \begin{cases} 1, \text{ if any cluster is allocated to node } i \\ 0, \text{ otherwise} \end{cases}$$

M_i = cost of node *i*

the model constraints are formulated as

$\exists i \cdot \Theta_{i,j} = 1 \Leftrightarrow \Omega_j = 1$
for all objects *j*

$\exists i \cdot \Psi_{i,j} = 1 \Leftrightarrow \Phi_j = 1$
for all clusters *j*

$$\sum_j \Theta_{i,j} = 1$$

for all objects *j*

$$\sum_j \Psi_{i,j} = 1$$

for all clusters *i*

which states that when an object is assigned to a cluster, there *is* an object at that cluster, etc. The model constraints also require every object to be assigned to only one cluster and every cluster to be assigned to only one node.

5.3 Optimization objectives

Without consideration of performance or dependability, a simple formulation is to minimize cost, i.e.

$$\text{min:} \sum_i M_i \Theta_i$$

subject to the constraints from section 5.2. Although performance as well as QoS requirements are ignored, this optimization objective is of some interest as it gives a lower bound on the cost of the system. Further, with $M_i = 1$, the result is the least number of computing nodes that is needed.

When minimization of capital cost is the objective, performance and QoS requirements are included as additional constraints in the optimization problem, i.e. *"What is the system structure that offers the least capital cost and which fulfils these performance and quality characteristics?"*

An alternative objective is optimization of performance and QoS, and with capital cost modelled as constraints, i.e. *"What is the system structure that offers the best performance and quality characteristics given this constraint on capital cost?"*

An objective function for process allocation which maximizes reliability is given in [42] which is readily adapted to object and cluster allocation.[12] Reliability is maximized when the mean number of failure is minimized:

$$\text{min:} \sum_n \sum_c \gamma^N_n \cdot \Psi_{c,n} \cdot e_c + \sum_l \sum_{(s,c,o) \in A_L(l)} \gamma^L_l \cdot m_o \cdot \lambda_{(c,s,o)} \cdot \Delta t / C^L_l$$

where γ is the failure intensity of nodes and links, e_c is the accumulated execution time of cluster *c* (on node *n*) during

[12] *The formulation in [42] is more general as each path between nodes can be constructed by a sequence of links. In our formulation, nodes are directly interconnected by (virtual) links.*

an interval Δt. The first part is the average number of node failures, while the second part is the average number of communication failures.

Minimization of waiting time[13] is expressed as the objective function:

$$\min: \sum_n \sum_o \frac{\lambda_{n,o}}{\Lambda} W_{n,o}$$

where

$$W_{n,u} = \frac{1/\mu_{n,u}}{1-U_n}$$

is the waiting time at node n for operation o, $\mu_{n,o}$ is the service rate of the operation, $\lambda_{n,o}$ is the activation frequency and Λ is the total workload on all nodes. Both the service rate and the utilization U_n depend on the number of clusters and objects allocated to the node.

5.4 Optimization techniques

A number of optimization objectives are presented in the previous subsection. Assignment of objects to clusters and clusters to nodes is at least as difficult as assignment of files to nodes or tasks to processors. These problems are known to be NP-hard [45], which implies that direct solution techniques from mathematical programming are not feasible. A number of heuristic solutions has been proposed for allocation of files and clusters in distributed databases [e.g. 11, 12, 45, 4, 39], including techniques based on

- knapsack
- branch and bound
- network flow.

Heuristic solutions of task assignment to processors have been studied [e.g. 34, 2, 42, 7]. As task assignment and file allocation are similar problems, many of the heuristic approaches are similar. In addition, techniques from artificial intelligence are used in [42], where the solution state space is examined with an adaptation of the A* algorithm.

Solutions to difficult problems like task assignment and object and cluster assignment, can be made much more effective with a good initial solution. The aggregated interaction frequency between client and server objects

$$\lambda_{(c,s)} = \sum_o \lambda_{(c,s,o)}$$

can be used to form clusters such that objects with frequent interaction are assigned to the same cluster (and objects with infrequent interaction are assigned to different clusters). However, the specification transformation resulting from assignment of interacting objects to different clusters, results in a modification of object interaction. It is not sufficient to just group objects with high interaction rates. In addition, the capacity of nodes, storage and links limit the number of objects and clusters that can be assigned to the same node. However, good initial solutions can be found, which limits the examined state-space and reduce the running time.

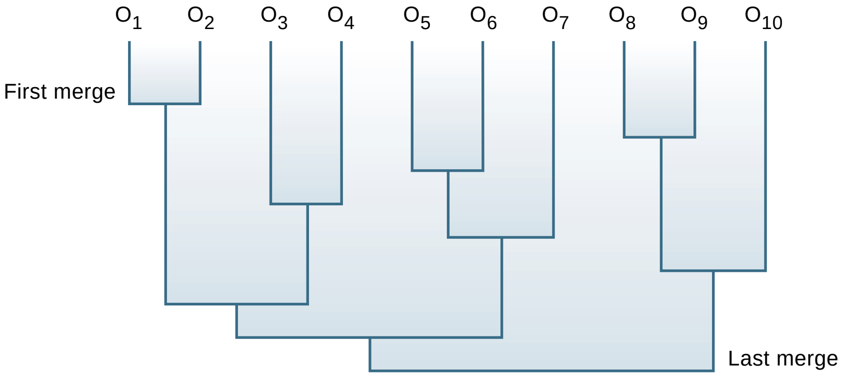


Figure 16 Hierarchical clustering

Cluster analysis is a collection of techniques used in many branches of science to group or classify entities. In this case, the term ‘cluster’ is different from ‘cluster’ as defined by RM-ODP, but is a general grouping of objects based on some measure of ‘dissimilarity’ or, equivalently, ‘similarity’ between objects. The result of cluster analysis depends both on the measure of dissimilarity and the algorithm to group objects (see [13, 28, 1]). Definition of the dissimilarity measure depends on the application domain, while clustering algorithms are applicable between domains. However, not all algorithms are equally suitable, and a clustering algorithm should be applicable in an application domain.

In general, clustering algorithms can be classified as hierarchical, overlapping and partitioning. Hierarchical clustering starts with each object allocated to one cluster. Following this, the two most similar clusters are merged until a single cluster is formed (or a predefined number of clusters is reached). Hierarchical clustering is often illustrated a tree as shown in Figure 16 where the height indicates when clusters are merged. Clusters formed by hierarchical clustering are disjoint at each level in the tree, e.g. each object is assigned to only one cluster. When engineering communicating systems, the hierarchy of clusters can be used to select objects to be assigned to ODP clusters, and to select objects to move to other ODP clusters when an object assignment violates constraints (e.g. capacity or QoS requirements).

Overlapping or fuzzy clustering is possible when similarity between objects is used to assign a degree of cluster membership to objects. Fuzzy clustering is frequently used within the fields of pattern recognition and data mining, but can also be applied to guide assignment of objects to ODP clusters.

While hierarchical and overlapping clustering start with individual objects and gradually merge objects to larger clusters, partitioning is a technique where a single cluster is divided into smaller clusters. Partitioning is generally useful when the result is few but large clusters. Fragmentation of tables in a distributed database is an example where partitioning can be used to group attributes into fragments that are distributed [45].

[13] *As only transactions are addressed, the throughput is in equilibrium equal to the applied workload.*

How to measure similarity $\delta_{(c,s)}$ between two objects *c* and *s* in a system? Both the invocation frequency between objects $\delta_{(c,s)} = \lambda_{(c,s)}$, as well as the average bandwidth requirement between objects

$$\delta_{(c,s)} = \sum_{o} m_o \cdot \lambda_{(c,s,o)}$$

can be used. Message size information is incorporated in the bandwidth requirement, and is most likely the best choice. However, when 'short' messages are used by prioritized applications, and 'larger' messages are less important, the invocation frequency can be the better choice of similarity measures.

Given a similarity measure, different hierarchical clustering algorithms (or strategies) are possible. A simple strategy is single-link clustering. In this case, clusters with the two most similar objects are joined, e.g. find two objects *c* and *s* such that

$$\delta_{(c,s)} + \delta_{(s,c)}$$

is maximized and join the clusters containing objects *c* and *s*. A drawback of this technique is that clusters can be constructed with little internal cohesion as only interaction between pairs of objects is used. This is avoided by complete-link clustering which joins clusters such that the similarity between all objects in the two clusters is maximized, e.g. joint clusters *A* and *B* such that

$$\sum_{\substack{a \in A \\ b \in B}} \left(\delta_{(a,b)} + \delta_{(b,a)} \right)$$

is maximized. Complete-link clustering maximizes internal interaction and message exchange, while interaction between clusters is minimized.

Given the objects from Figure 16, a heuristic approach initially assigns all objects to a single cluster on a single node. Assuming system constraints are violated, the cluster is split by reversing the last merge operation. Hence, two clusters with objects {O_1, O_2, O_3, O_4, O_5, O_6, O_7} and {O_8, O_9, O_{10}} are formed and assigned to different nodes. In this case, the transformation rules from section 3.3 are applied to augment the object model with necessary distribution support objects. If constraints are still violated, the cluster last merged is split. In this case, the larger cluster {O_1, O_2, O_3, O_4, O_5, O_6, O_7} is split into {O_1, O_2, O_3, O_4} and {O_5, O_6, O_7}. This process is repeated until all constraints are satisfied (if possible).

The method outlined above is not guaranteed to produce an optimal solution, as many possible configurations are not investigated. However, clusters are split such that the most similar objects are kept together, which is good engineering practice.

6 Conclusion

An overview of engineering principles for communicating systems has been presented. Optimal allocation of computational viewpoint objects to engineering viewpoint clusters is an NP-hard problem and heuristic solutions are necessary. As many different distribution schema are possible, system specifications express requirements on the distributed processing environment as QoS attributes associated with functional capabilities. When a particular distribution schema is selected, QoS attributes are used to guide specification refinement. The refinement process augments the specification with the necessary distribution support objects. Clustering of objects based on object interaction and the bandwidth requirement between objects are proposed to help select useful distribution schema. Each schema can be analyzed, using analytical techniques from queuing theory, to select the schema that is optimal given an objective function and a set of system constraints.

7 References

1 Arabie, P, Hubert, L J, De Soete, G (ed). *Clustering and classification.* Singapore, World Scientific Publishing, 1996.

2 Bowen, N S, Nikolaou, C N, Ghafoor, A. On the Assignment Problem of Arbitrary Process Systems to Heterogenous Distributed Computer Systems. *IEEE. Trans. Comput.,* 41 (3), 275–73, 1992.

3 Buzen, J P. Computational algorithms for closed queuing networks with exponential servers. *Comm. ACM,* 16 (9), 527–31, 1973.

4 Ceri, S, Martella, G, Pelagatti, G. Optimal File Allocation in a Computer Network : a Solution Method Based on the Knapsack problem. *Computer Networks,* 6, 345–57, 1982.

5 Chapman, M, Montesi, S. *Overall Concepts and Principles of TINA.* TINA Consortium, 1995. (Technical report.)

6 Chu, T C K, Abraham, J A. Load Balancing in Distributed Systems. *IEEE Trans. Soft. Eng.,* SE-8 (4), 401–12, 1982.

7 Chu, W W et al. Task Allocation in Distributed Processing. *IEEE Computer,* 13 (11), 57–69, 1980.

8 Cristian, F. Understanding fault-tolerant distributed systems. *Communications of the ACM,* 34 (2), 57–78, 1991.

9 Dallery, Y, Cao, X R. Operational analysis of stochastic closed queuing networks. *Performance Evaluation,* 14, 43–61, 1992.

10 Denning, P J, Buzen, J P. The operational analysis of queuing network models. *Comput. Surveys,* 10 (3), 225–61, 1978.

11 Dowdy, L W, Foster, D V. Comparative Models of the File Assignment Problem. *ACM Comput. Surv.,* 14 (2), 287–313, 1982.

12 Eswaran, K P. Placement of Records in a File and File Allocation in a Computer Network. In: *Information Processing '74.* Stockholm, 1974, 304–307.

13 Everitt, B R. *Cluster analysis.* Halstead Press, 1993.

14 Fabre, J-C, Prennou, T. Friends : A Flexible Architecture for Implementing Fault Tolerant and Secure Distributed Applications. In: *Second European Dependable Computing Conference (EDCC-2).* Taormina, Italy, 1996.

15 Ferrari, D. *Computer Systems Performance Evaluation.* Prentice-Hall, 1978.

16 Franken, L. *Quality of Service Management : a Model-Based Approach. Performability modelling tools and techniques.* Centre for Telematics and Information, University of Twente, 1996. (PhD thesis.)

18 Graham, R L, Knuth, D E, Patashnik, O. *Concrete mathematics,* Addison-Wesley, 1989.

19 Harrison, P, Patel, N. *Performance Modelling of Communication Networks and Computer Architectures.* Addison-Wesley, 1993.

20 Haverkort, B R, Niemegeers, I G. Performability modelling tools and techniques. *Performance Evaluation,* 25, 17–40, 1996.

21 Hughes, P. *Performance Engineering.* Trondheim, Department of Computer Systems and Information Science, Norwegian University of Science and Technology, 1997. (Technical Report.)

22 ISO/IEC JTC 1/SC 21/WG7 N1192. Working document on QoS in ODP. 1997.

23 ITU. *Principles for a Telecommunications management network.* Geneva, ITU, 1996. (ITU-T Recommendation M.3010 (05/96).)

24 ITU. *Security architecture for Open Systems Interconnection.* Geneva, ITU, 1991. (ITU-T Rec. X.800.) (ISO 7489-2, 1989.)

25 ITU. *Open Distributed Processing : Reference Model : Foundations.* Geneva, ITU, 1995. (ITU-T Rec. X.902.) (ISO/IEC 10746-2, 1996.)

26 ITU. *Open Distributed Processing : Reference Model : Architecture.* Geneva, ITU, 1995. (ITU-T Rec. X.903.) (ISO/IEC 10746-3, 1996.)

27 ITU. *Open Distributed Processing : Trading Function, Part 1 : Specification.* Geneva, ITU, 1996. (Draft ITU-T Rec. X.950.) (ISO/IEC DIS 13235-1. Geneva, 1996.)

29 Jain, A K, Dubes, R C. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

30 Kim, C, Kameda, H. An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems. *IEEE Trans. Comput.,* 41 (3), 381–84, 1992.

31 Laprie, J-C (ed). *Dependability : Basic Concepts and Associated Terminology.* Springer, 1992. (PDCS Tech. Rep. No. 31, May 1990.)

32 Lazowska, E D et al. *Quantitative System Performance : Computer Systems Analysis Using Queuing Network Models.* Prentice-Hall, 1984.

33 Linington, P. RM-ODP : the architecture. In: *3rd international IFIP TC6 Conference on Open Distributed Processing.* 1995, 15–33.

34 Lo, V M. Heuristic Algorithms for Task Assignment in Distributed Systems. *IEEE Trans. Comput.,* 37 (11), 1384–97, 1988.

35 Menascé, D A, Almeida, V, Dowdy, L W. *Capacity Planning and Performance Modeling.* Prentice-Hall, 1994.

36 Meyer, J F. On evaluating the performability of degradable computing systems. *IEEE Trans. Comput.,* 29 (8), 720–31, 1980.

37 OMG. *The Common Object Request Broker : Architecture and Specification.* 1995. (Technical report, revision 2.0.)

38 Opdahl, A L. *Performance engineering during information system development.* Trondheim, Department of Computer Systems and Telematics, Norwegian Institute of Technology, 1992. (Dr.ing (PhD) thesis.)

39 Ramamoorthy, C V, Wah, B W. The Isomorphism of Simple File Allocation, *IEEE Trans. Comp,* 32 (3), 231–231, 1983.

40 Raymond, K. Reference model of open distributed processing (RM-ODP) : Introduction. In: *3rd international IFIP TC6 Conference on Open Distributed Processing,* 1995, 3–14.

41 Rumbaugh, J et al. *Object-oriented modeling and design.* Prentice-Hall, 1991.

42 Shatz, S. Task Allocation for Maximizing Reliability of Distributed Computer Systems. *IEEE Trans. Comput.,* 41 (9), 1156–68, 1992.

43 Sienknecht, T, Martinka, J, Friedrich, R. Murky transparencies : Clarity using performance engineering. In: *3rd international IFIP TC6 Conference on Open Distributed Processing,* 1995, 507–510.

44 Smith, C. *Performance Engineering of Software Systems.* Addison-Wesley, 1989.

45 Tamer Orzu, M, Valduriez, P. *Principles of Distributed Database Systems.* Prentice-Hall, 1991.

46 Williams, H P. *Model building in mathematical programming.* Wiley, 1993.

Knut Johannessen is Senior Engineer at Telenor Nett, IT department, working with strategy for operational support systems and IT architecture. He is participating in ITU-T SG4 which specifies Recommendations for TMN. He is also currently pursuing a Ph.D. at the Department of Telematics, Norwegian University of Science and Technology with focus on architectural design of distributed systems.

e-mail: knut.johannessen@s.nett.telenor.no

The TINA Architecture

TOM HANDEGÅRD AND LILL KRISTIANSEN

TINA is a general software architecture, applicable to a broad range of telecommunications and information services. The architecture is based on the principles of object-orientation, CORBA-based distributed processing, and separation of services from underlying network. The purpose is to provide for common services over different underlying network technologies and rapid service development in an open environment that encourages third party development. This paper provides an overview of the TINA architecture and contrasts it with related work in the area of multimedia services, intelligent networks, and broadband networks.

1 Introduction

The Telecommunications Information Networking Architecture Consortium (TINA-C) consists of approximately 40 telecommunications and information technology companies from all over the world. The consortium started in 1993 with the goal to define and validate a software architecture that will enable efficient introduction and management of new and sophisticated telecommunications services. The TINA architecture is based on object-oriented technology and distributed computing, and incorporates results from international standards such as ITU/ISO RM-ODP, ITU-T TMN, ATM Forum, and OMG's CORBA architecture.

During the first phase of TINA-C, extending from 1993 to 1997, the TINA architecture was developed by a Core Team of about thirty member company engineers located in Bellcore's premises in New Jersey, USA. The basic set of TINA specifications are now considered stable, and are available to the public through its Web site (http://www.tinac.com). The second phase of the consortium is now starting, and is planned to extend for three years through the year 2000.

The TINA architecture has three major goals [1]:

- To make it possible to provide versatile *multimedia services*
- To make it easy to *create* and *manage* services and networks, and
- To create an *open* telecommunications software component marketplace.

TINA is intended to be a general software architecture, applicable to a broad range of telecommunications and information services. It is, however, specifically intended to suit the following areas of application: New broadband networks (ATM networks and B-ISDN), new sophisticated services (multimedia services, multiparty conferencing, VPN services, and IN-type services), and nomadic communication and new mobile networks (uniformly providing user mobility, terminal mobility and session mobility, UMTS, and DECT). Note that TINA was planned and initiated *before* the spectacular raise in Internet and Intranet technology during the recent years. However, the TINA work has not been unaffected by the Internet developments, and many of the TINA results (e.g. the Service Architecture) are, due to their quite general nature, seen as useful in the context of the Internet.

The TINA architecture is based on the principles of *object-orientation, distribution*, and *separations of concern*. The purpose of these principles is to ensure interoperability, portability and reusability of software components, independence from specific technologies, and to share the burden of creating and managing a complex system among different business stakeholders, such as users, service providers and network providers.

Object-oriented techniques focus on reducing complexity through modularization, encapsulation of data and methods, and reuse of existing components. Breaking a complex system down into a set of encapsulated objects reduces complexity, and de-couples the software modules from each other so that a change in one component due to a change in underlying technology, (standards, languages, programs, networks, etc.) does not affect other components.

Distribution of service software components over different parts of the network can improve performance by reducing network load and applying load balancing techniques. Furthermore, services may be made more fault tolerant by the application of techniques such as object replication. Finally, distribution is to some extent an inherent element of many telecommunication services. A key principle for TINA is that telecommunications services and management systems are considered as software applications that operate in a distributed environment. To simplify the development of these distributed applications TINA mandates the use of a Distributed Processing Environment (DPE). The purpose of the DPE is to hide from the applications the details of the underlying technologies and distribution concerns, thus simplifying the construction of distributed software.

TINA adopts two major *separations of concern*, see Figure 1. The first separation is between applications and the underlying distributed processing environment (DPE) on which they run, as described in the previous paragraph. The second is the separation of applications into a service-specific part and a network control and management specific part. The latter interacts with the transport network. The separation into a service part and a network part is one of the key features of the TINA architecture. It allows provision of common services for different underlying network technologies, including SDH, ATM, and radio networks.

According to the separation principles, TINA is divided into three sub-architectures: The *Computing Architecture* [2, 3, 4] defines the DPE and associated modelling concepts. The TINA DPE is based on OMG's CORBA with adaption

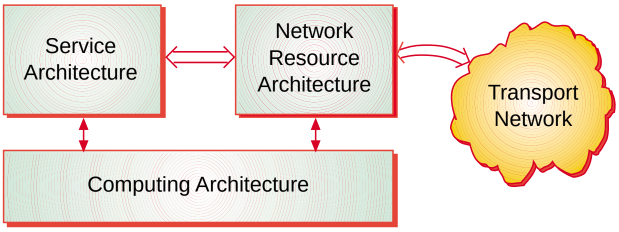


Figure 1 The TINA architecture consists of three sub-architectures: Computing, service, and resource architectures

for telecommunications requirements. The *Service Architecture* [5] defines a set of principles for providing services. It uses the notion of session to offer a coherent view of the various events and relationships taking place during the provision of services. The *Network Architecture* [6] describes a generic, technology independent model for setting up connections and managing telecommunication networks.

The TINA Service and Resource Architectures are based on a specific model of how business within the telecommunications domain will be conducted. The TINA *Business Model* [7] describes the different parties involved in service provisioning and their relationships to each other. A small number of roles are defined, which reflect the major business separations of a complex telecommunications and information market: Consumer, retailer, broker, third party service provider, content provider, and network provider. Within the service and resource architectures, Reference Points are identified. Each reference point comprises a set of interfaces describing the interactions taking place between these roles. For the most important reference points, detailed specifications have been developed [8, 9, 10]; other interfaces are yet to be described in detail.

2 Overview of the TINA Architecture

This chapter provides an overview of the different parts of the TINA architecture, and how they fit together. It is important to note that in a telecom system, some parts may conform to TINA, whilst other parts may not. For instance, it is possible to build services that conform to the TINA service architecture on networks that are not consistent with the TINA resource architecture. In this case, the system uses the TINA service architecture, but not the TINA resource architecture.

2.1 The Computing Architecture and DPE

The TINA Computing architecture consists of three parts: Information Modelling Concepts [2], Computational Modelling Concepts [3], and DPE Architecture [4]. The TINA computing architecture is based on ITU/ISO's Reference Model for Open Distributed Computing (RM-ODP), and the three parts corresponds roughly to RM-ODP's information, computational, and engineering viewpoints, respectively.

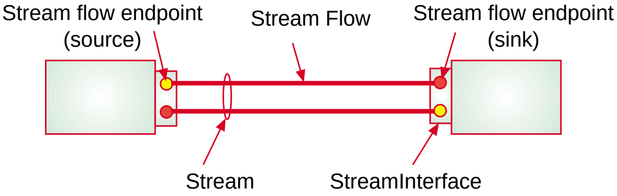


Figure 2 Stream interfaces and stream flows

The information and computational modelling concepts define the concepts and languages that are used to write specification of TINA Reference Points and software components. The information and computational parts of TINA specifications are considered complementary to each other.

The information model describes the information-bearing entities in the application (or component, or reference point), their relationships to each other, and the constraints and rules governing their behaviour. It identifies the entities that the application deals with and is independent from distribution concerns. The language used for TINA information specifications is Object Modelling Technique (OMT) [12] for graphical specifications, and a simplified version of GDMO for textual specifications.

The computational model focuses on the software modules, objects, and their interfaces. Applications consist of collections of objects that interact with each other via interfaces. Each object provides one or more interfaces to enable other objects to access its capabilities.

Objects interact either by invoking operations and sending responses or by means of *stream flows*. Interfaces used for the former kind of interaction are called *operational interfaces*, and interfaces used for the latter kind are called *stream interfaces*. The interactions that occur at an operational interface is based on the paradigm of remote procedure calls, i.e. they are structured in terms of invocations of one or more operations and responses to these invocations.

A stream interface is an abstraction that represents an end-point of a continuos information flow, such as a video or audio flow. When objects interact via stream interfaces, the information exchange occurs in the form of stream flows between the objects, where each stream flow is unidirectional and is a bit sequence with a certain frame structure (data format and coding) and quality of service parameters.

The computational objects are specified using TINA-ODL (Object Definition Language). TINA-ODL is a pure superset of OMG IDL. The major part of a typical ODL specification of a software module is description of the attributes and operations it supports, using OMG IDL.

The CORBA architecture is the basis for the TINA DPE. The CORBA architecture does not meet *all* the requirements of telecommunications applications, however. Consequently, TINA-C's work within the DPE area has been focused on the identification of additional requirements and working within the OMG to have the current standards extended or new standards established. Extensions that have been addressed by TINA include multimedia support (control of audio/video streams), support for real-time requirements, support for alternative transport protocols (other than TCP/IP) such as SS7 and ATM, extensions to IDL, and new object services (e.g. TMN-style notification service). Refer to [11] in this issue of *Telektronikk* for more information on CORBA and the OMG.

The DPE (Figure 3) is a layer on top of the underlying operating system and communications software. The DPE nodes have basic communication capabilities to set up connections between them in order to establish communication channels that can be used to carry operational communication between the application objects. These communication capabilities are in the TINA terminology referred to as the *kernel transport network* (KTN). The KTN can be compared to the signalling network in traditional telecommunication system architectures.

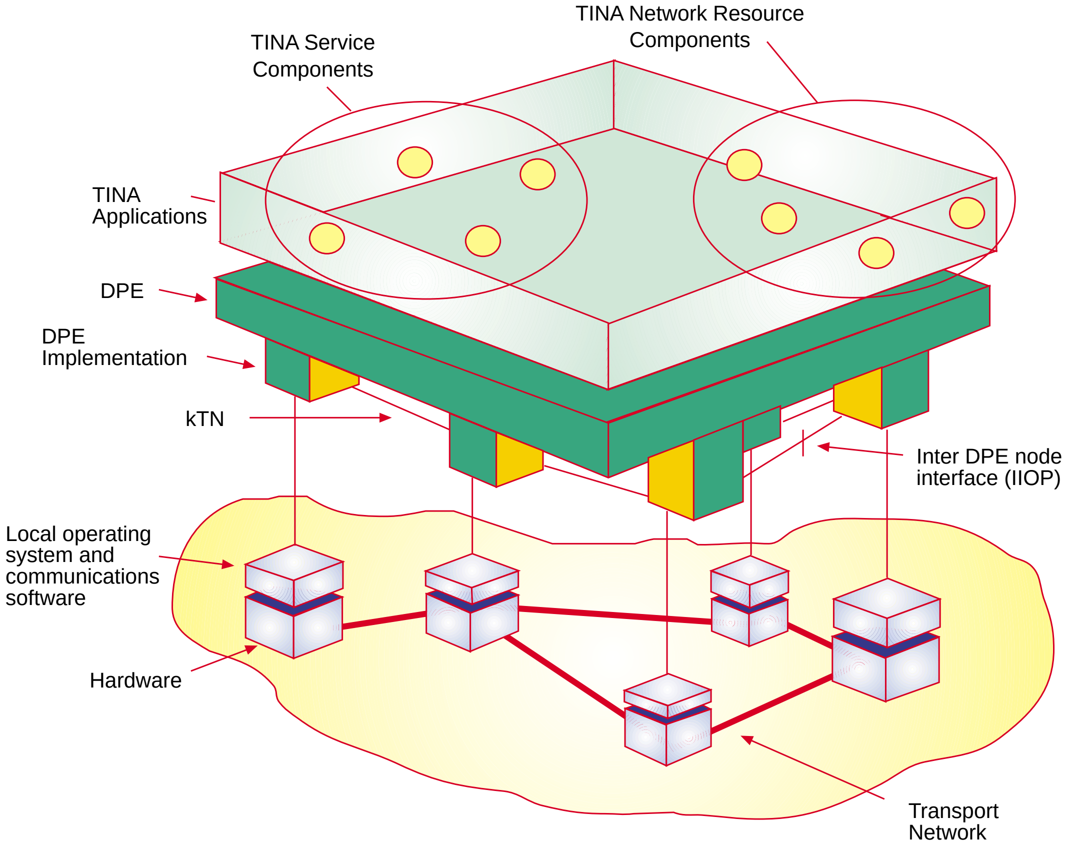

TINA Network Resource Components
TINA Service Components
TINA Applications
DPE
DPE Implementation
kTN
Inter DPE node interface (IIOP)
Local operating system and communications software
Hardware
Transport Network

Figure 3 TINA DPE and kernel transport network

2.2 Business Model and Reference Points

The TINA business model and reference points [7] consists of two parts:

1. A generic framework for specifying interactions and contracts between business administrative domains. This includes templates and languages for informational and computational specifications.
2. A set of defined business roles, and for each defined relationship between roles, definition of reference points.

The generic framework is important to allow for openness and flexibility. It allows two business administrative domains to define their own business roles and interactions.

The defined business roles and reference points identify components and functionality that are specific to TINA. The business roles have been identified by analysing the current business relationships in telecommunication and information services.

2.2.1 TINA Business Roles

The identified set of TINA business roles are (see Figure 4):

- *Consumer* – a stakeholder that consumes services, e.g. a private person, a household or a company. This type of stakeholders is the ultimate target for the business in a TINA system. All other types of stakeholders can be characterised as “producers” or “middlemen”.
- *Retailer* – a stakeholder that provides (sells) different kinds of services to the consumers. Retailers ensure ease of access and provide quality guarantees to consumers.
- *Third Party Service Provider* – a stakeholder that provides services to stakeholders other than consumers.
- *Broker* – a stakeholder that provides information about how to find services and stakeholders in the TINA system. A broker provides a specific service, which has a general use in a TINA system.
- *Network Provider* – a stakeholder that provides transport services and controls and manages communication equipment, like switches, cross-connects, bridges, routers and trunks. Note that the term ‘network provider’ must be understood in a strict sense, not as a synonym for Public Network Operators (PNO). The PNOs will to a large extent operate also in the retailer business role.

It is important to notice that one *business administrative domain*, such as a PNO, may simultaneously perform several of the defined business roles, and internally it might either comply with the defined reference points, or use other proprietary interactions.

The main activities of the *consumer business role* are:

1. Obtain location of retailers, service providers, and other consumers.
2. Initiate service relationships that include service providers and other consumers.

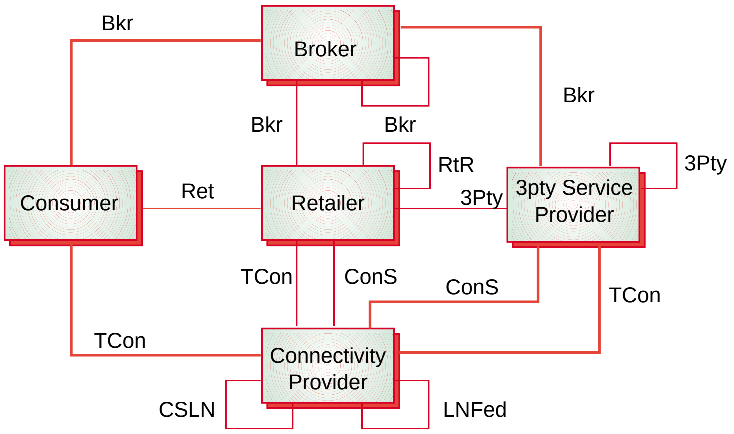


Figure 4 TINA business roles and reference points. The reference points defined are called Retailer RP (Ret), Third-party RP (3Pty), Connectivity service RP (ConS), Client-server layer network RP (CSLN), Layer Network Federation RP (LNFed), Retailer-to-Retailer RP (RtR), Terminal Connectivity RP (TCon), and Broker RP (Bkr)

3. Register and de-register at retailers.
4. Indicate availability to retailers, e.g. for receiving invitations and calls.
5. Accept downloads from retailers to upgrade the interaction capability with the retailer. This allows tailored, non-standardised services to be provided by the retailer, and makes *tailored services* and service functionality an issue in the *competition* between retailers.

Consumers can use one or more retailers, even at the same time. The life-time of a relationship between a particular consumer and a particular retailer can vary from seconds to years. For example, a consumer might use a particular service offered by a retailer only once. Another example is a consumer that is a faithful client to a particular retailer for a long time and uses the retailer as a "one-stop-shop".

The main objectives for the *retailer business role* are the following:

1. Provisioning and management of services
2. Collecting accounting information for the purpose of billing for service usage
3. Establishing relationships to other service providers, both other retailers and third party providers
4. Value adding services from third party providers.

A retailer can deploy a new service for immediate use by any consumer in the TINA system *without consulting or standardising the services with other retailers*. This is an absolute requirement that will allow a TINA system to be an attractive and dynamic system for the future. Together with the downloading capability of the consumer this enables rapid service deployment.

A retailer can interact with other retailers and/or third party service providers. Often the term 'service provider' is used for both. The difference between these two roles may not be so big, but a retailer has his own subscribers, while a third party service provider does not. The third party provider may be a pure content provider, providing content to be distributed by the retailer to the subscribers of the retailer.

A stakeholder in the business role of *network provider* controls and manages a transport network containing switches, cross-connects, routers and trunks. The transport network is controlled by TINA Connection Management, see section 2.4.

The transport network operated by a single network provider is unlikely to be a global network that connects all the consumers, retailers and third party service providers. The global transport network is most likely to be segmented into a number of subnetworks controlled by different stakeholders, each in the network provider business role. To allow management (set-up, removal, etc.) of connections routed through two or more network segments belonging to different connectivity providers, the connectivity providers have to federate. The same principles that apply to NNI (network-to-network interface) for bearer services in traditional networks apply to federation among connectivity providers. The reference point LNFed deals with the interface between federated subnetworks operated by different network providers.

TINA supports the concept of layer networks. A client/server relationship exists between the connectivity provider that manages the client layer network and the connectivity provider that manages the server layer network. The client layer network uses resources of the server layer network. The reference point CSLN deals with the interface between client and server layer networks operated by different network providers.

Stakeholders in the *broker business role* have a specific mission in the TINA system, which is to provide stakeholders with information on how to reach other stakeholders and business administrative domains and how to reach services provided in the TINA system. A broker may act as an entity that ensures equal access to services. It may also act much the way an Internet search engine does. The main objectives of this business role are:

1. Obtain location of retailers
2. Register and de-register at retailers
3. Indicate availability to retailers.

2.2.2 TINA reference points

In order for a system to become TINA compliant, some conformance requirements have to be fulfilled. The conformance requirements are expressed in terms of reference points. If one or more of the defined reference points are fulfilled, the system is a TINA system. Two types of conformance requirements are relevant to TINA:

1. **Interdomain reference points:** Conformance requirements for interoperability between different business administrative domains
2. **Intradomain reference points:** Conformance requirements for TINA components developed by different vendors and to be used within one administrative domain.

The interdomain reference points serve the same purpose as NNI interfaces in B-ISDN signalling or X reference points in TMN. They deal with interfaces between different administrative domains.

The intradomain reference points may be compared to e.g., SSP-SCP INAP in IN and Q interfaces in TMN. They deal with component interworking and mainly represent requirements on vendors. The details of intradomain reference points are not currently worked out in TINA, but the service and network component specifications [13, 14] will contain good candidates for such reference points.

Each interdomain reference point has two parts: *Access part* and *usage part*.

The *access part* contains the interfaces that are needed to establish a contractual relationship over an administrative domain boundary. TINA has attempted to define a generic access part that can be as much as possible applicable to all reference points. This may actually not be completely achievable because different requirements apply to different reference points. For example, the access part between the consumer and the retailer will typically be asymmetric, with the retailer providing access to the consumer, while the relation between two network providers may be peer-to-peer and symmetric. The security issues may also vary. However, the notion of access session is useful in all cases where administrative domains are crossed. Currently, the asymmetric access part is defined. Its definition is placed within the Ret reference point document [8].

Note that the separation of access and usage parts allows the access to be conformant to the TINA reference point, while the usage part can be otherwise agreed between the involved business administrative domains in case the TINA defined usage parts do not fit these business administrative domains.

The *usage part* depends completely on the business roles involved and the services to be performed.

Specifications are currently completed for the Ret, TCon, and ConS reference points. These reference points are defined in detail, including CORBA IDL definitions, message sequence charts and information models.

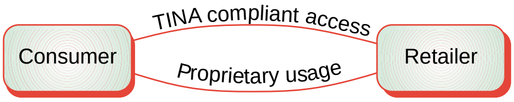


Figure 5 Separation of access and usage allows a standard protocol to be used for access, while a proprietary protocol is used for service usage

The Ret specification is the most complex one. Ret contains the asymmetric access part, and all the necessary interfaces and information to perform the service session control between one retailer and one or several consumers. The Ret specification also includes the operations needed at communication session level. Thus, Ret contains specifications for most of what is described in the TINA Service Architecture. Great care has been taken to make the specifications well structured and modular. In fact, the specification has been divided into modules, called *feature sets*, that to a large extent can be used individually. The specifications may be worth looking into for anyone planning to build large IDL specifications, even if they are not specifically interested in the particular information handled over Ret.

ConS and TCon are also fully defined specifications, though not that complex. TCon needs to be supplemented with details for each technology involved.

It is likely that a lot of work done for the currently defined reference points can be reused for the remaining reference points.

2.3 Service Architecture

2.3.1 Requirements

The TINA Service Architecture is designed to achieve:

- Support for a wide range of services, including IN-like services like ‘freephone’ and queuing services, and also multimedia, multiparty services with all possible payment and service control schemas
- Support for rapid service development, including development of services that are not standardised
- Support for tailored services
- Support for a multiplayer environment, e.g. to allow equipment and software from different vendors within one business domain, and also to allow several service domains to offer compound services
- Support for service manageability, including but not limited to customer access to management services, e.g. management of timetables to determine personal availability, or on-line management of subscriptions
- Universal service access, including global mobility, such as personal mobility and service mobility
- Ubiquitous information access, e.g. support to find information even when the information server has moved to another place.

TINA does not standardise specific services like ‘Video On Demand’ or ‘Tele education’. Instead, the TINA Service Architecture provides a framework that supports personal mobility, multiparty services and service tailoring. The Service Architecture covers the upper part of Figure 4, including the relationships between consumers, retailers, brokers and service providers.

2.3.2 Service Architecture Basics

The Service Architecture is illustrated in Figure 6. The picture is somewhat simplified. The objects involved are explained as follows:

Initial Agent (IA), *User Agent* (UA) and *Provider Agent* (PA) are the objects related to the access part. The binding between UA and PA is called an *access session*. A session may be seen as a collection of objects collectively fulfilling a task.

The IA exists due to initial, insecure access, and it also provides anonymous access. *Access specific UAP* (as-UAP) may be seen as a presentation tier towards the end user.

After the logon procedure has established an access session, the user may typically either start a service of his own, or join

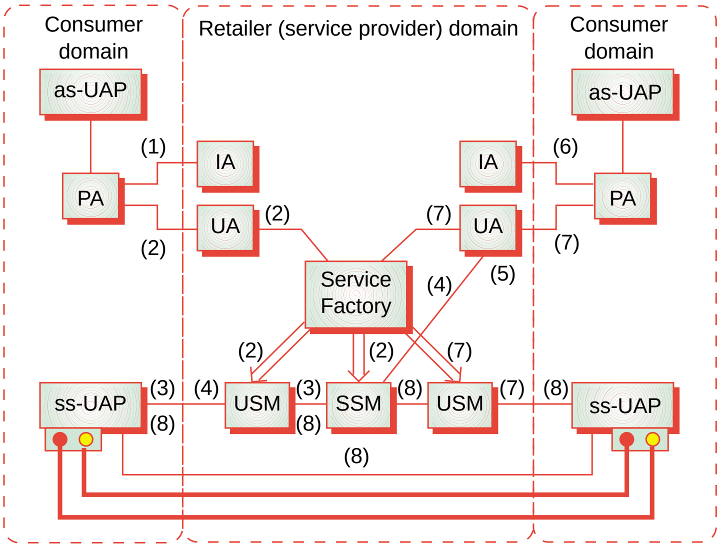


Figure 6 Example of objects related to the service architecture

an existing service session, as a reply to an invitation. The invitation mechanism is the linkage between the access session and the service session. There are also relations to ensure that the service sessions controlled by an access session will die when the access session dies.

The main objects related to service usage are *Service Session Manager* (SSM), *User Session Manager* (USM), and *User Application* (UAP). The SSM is the core of the service; it contains the core service logic. Different types of SSMs will exist for different service types. The main purpose of the USM is to provide tailoring of services for specific users. The USM is optional, however. In case tailoring is not an issue, it may be omitted. Other service specific support objects, not shown in the figure, might also exist. In the retailer domain this may include mixing bridges (MCU) and other converters. The collection of SSM and USMs collectively controlling a service execution constitutes a *service session*.

The UAP may be service specific and contain service logic and user interface related functionality. The UAP may also be more generic, like a Web browser.

In case streams are needed as part of the service, e.g. for a Video-on-demand service, a communication session will be initiated.

Details of each of the three sessions – the access session, the service session and the communication session – will follow, but first the example scenario illustrated in Figure 6 will be explained. The scenario is as follows:

(1) Interactions to establish an access session.

(2) User 1 starts a service session, and the SSM is instantiated.

(3) User 1 uses the service as a single user information retrieval service, e.g. to browse the video offers of the day.

(4) User 1 invites user 2 to join his service session.

(5) The user agent for user 2 stores the invitation, e.g. because user 2 is currently not logged on, or has activated ‘do not disturb’.

(6) User 2 logs on.

(7) User 2 joins the service session; this may be done from the same terminal, or User 2 might use another access session from another terminal.

(8) The users exchange video interests or other service specific information. They also exchange more generic information like available stream interfaces, etc. They might possibly negotiate the payment schema, unless they choose a default schema. The necessary stream bindings are then set up through the communication session (not shown in this figure).

Some comments related to the relationships between the different sessions might be appropriate. A service session may be started from (i.e. controlled by) an access session of a user. A service session may also be started and controlled by the retailer. The former is the normal choice for a multimedia multiparty conference, the latter is well suited for chat-like services.

Some service sessions may be openly announced (like ‘chat-rooms’), while other service sessions are closed. For the openly announced sessions, end users may join freely, though they may still be charged for the service usage. For the closed sessions, the only way to even know the existence of the service session, is through receiving an invitation to join.

One service session might use and control several communication sessions. This is allowed for maximum flexibility. The service session might even hand over the control of a communication session to another service session, if appropriate.

Note that the pricing for the usage of the service is independent from the inviter/invitee distinction. Both ‘freephone’, ‘premium charge’ or ‘split charging’ may be in place at the service usage level. For more advanced multiparty services the payment schemas may be quite complex, e.g. the ‘boss’ pays for audio, those who wants video pay for themselves, or any other payment schema one can imagine.

2.3.3 The Access Session and user mobility

The access session related object *User Agent* (AU) has an important role in providing personal mobility and session mobility. The UA acts as a personal agent for a particular end-user. Upon reception of an invitation to join a service session – this may be anything from a simple point-to-point call or a complex multi-party-conference – the UA handles the invitation. The UA may either:

- Send the invitation to a selected terminal, e.g. depending on the type of the service
- Re-issue the invitation to someone else, e.g. the called user’s secretary
- Screen the invitation and ignore it
- Store the invitation, e.g. if the user is not logged on, his terminal is switched off, or he has activated ‘do not disturb’ functionality

- Start a service to handle the invitation in more sophisticated ways, e.g. the invitation may be addressed to "BigCompany", and based on location, type of service. etc., the invitation may be handed over to the appropriate service office, much like UAN services in IN.

Note that the invitation may be sent to one terminal, like a pager or another small terminal, while joining of the service session might be done from another terminal, e.g. if the service needs a terminal with a high quality screen.

TINA handles invitations and user mobility as part of a generic access mechanism, and not as a particular service. Remember also that the TINA access session may be used together with arbitrary services. This allows the personal mobility and access to be provided by TINA, even for services that are proprietary and do not use any of the generic TINA defined interfaces in the service session part, see Figure 5.

2.3.4 Service Session

The service session controls and manipulates a service instance. TINA has defined a number of generic service session components and interfaces. These generic components must be specialised or combined with service specific components to form specific services. The generic interfaces include:

1. Functionality to negotiate regarding transport connection requirements among the involved parties in a service session. Examples of parameters are quality of service requirements, supported protocols, and coding formats.
2. Generic control and voting mechanisms for multiparty services. This enables one party to propose a change in the service session, and the other parties to vote over this proposal before it is realised. The proposed change in the service session may be things like: adding another party, establish or change a video or audio connection, etc.

Service specific behaviour is of course totally dependent on the type of service. For a game service the service specific behaviour might check that the rules of an interactive game are followed. For a video on demand service specific operations related to content, like browse, rewind, stop, etc. fall into this category. Specialisation of the generic control and voting mechanisms mentioned above, e.g. to handle the different parties in a multi-party conference differently, is also possible.

The *Service Session Manager* (SSM) represents the core service logic. Different *User Service Session Managers* (USM) represent the different customisation of the service to the different end users. The *User Application* (UAP) represents the end user in the end user domain.

Tailoring can be done in several dimensions. It can be done by the USM, as explained above. Alternatively, it can be done by specialising the service, i.e. SSM, and optionally USM and/or UAP, by adding new interfaces and/or specialising the behaviour of the objects.

The UAP may be service specific and downloaded or otherwise provided from the service provider. It may be the case that the same UAP is used for many services, like a web browser. But in other cases the UAP may be specific for a particular service.

2.3.5 Communication session

In addition to the Access and Service Sessions described above, the TINA service architecture also defines the concept of *Communication Session*. Simply stated, the communication session provides a high-level view of underlying transport networks. More specifically, the overall goals of the communication session are to:

- Offer a technology independent view of the network to the service session
- Control and manage quality of service, set-up, modifications, etc. of multiple connections.

Some reasons why the communication session is separated from the service session are:

- It enables third-party control of connections. For example, some users participate in multimedia conferences using e.g. video streams, but without receiving the video streams themselves. Example: A conference manager may control and pay for some stream flows between some end users, but without participating in all stream flows himself.
- It enables the same communication session entities to be used by different services, i.e. the service tailoring may be done at the service level without affecting the behaviour of the communication related objects.
- It enables a service session to be *suspended*, and the communication session to be ended. On service *resumption* a new communication session can be started.

Figure 7 shows the objects making up the communication session, and their relationship to the service session, represented by the SSM (possibly USM) and the UAP(s).

As seen from the service session, the communication session takes care of the connections end-to-end between user applications (UAPs). The communication session related objects are *Communication Session Manager* (CSM) and *Terminal Communication Session Manager*

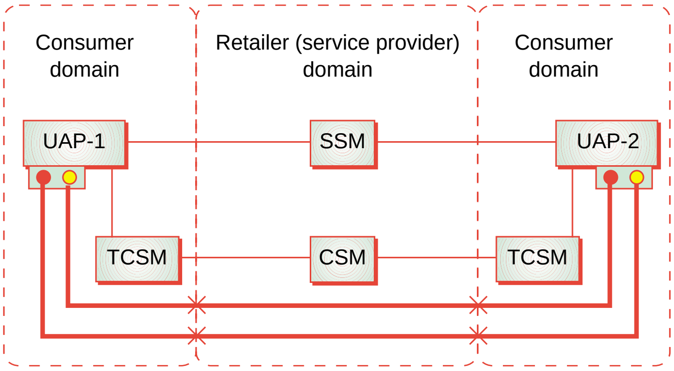


Figure 7 Objects involved in the Communication Session

(TCSM). These objects are further described in section 2.4.

The TINA Service Architecture assumes that the network is capable of informing the services about all the necessary events from the network. For example, for a video-on-demand service where the customer pays on-line for the video content, it is very important that the content is actually delivered on-line according to the agreement, if not, the price should be adjusted.

2.4 Network Resource Architecture

The TINA Network Resource Architecture (NRA) defines a generic technology independent framework for connection control (setting up, modifying and releasing connections) and for managing telecommunications networks. It consists of two parts, an information model called Network Resource Information Model (NRIM) and a computational part, the latter identifying software components and interfaces between them.

The NRIM is based on concepts from ITU-T standards. It extends these concepts to support different network technologies. The NRIM is intended to be general and network technology independent, yet it is a fact that the work has been focused on ATM networks.

An overview of the computational specification is shown in Figure 8. The NRA has three layers:

- The Communication Session layer
- The Connection Session Layer
- The Layer Network.

The *Communication Session* is the topmost layer. It provides service software with a simple, network independent interface to control and manage connections, called *flows* at this level. The flows extend between *flow termination points*.

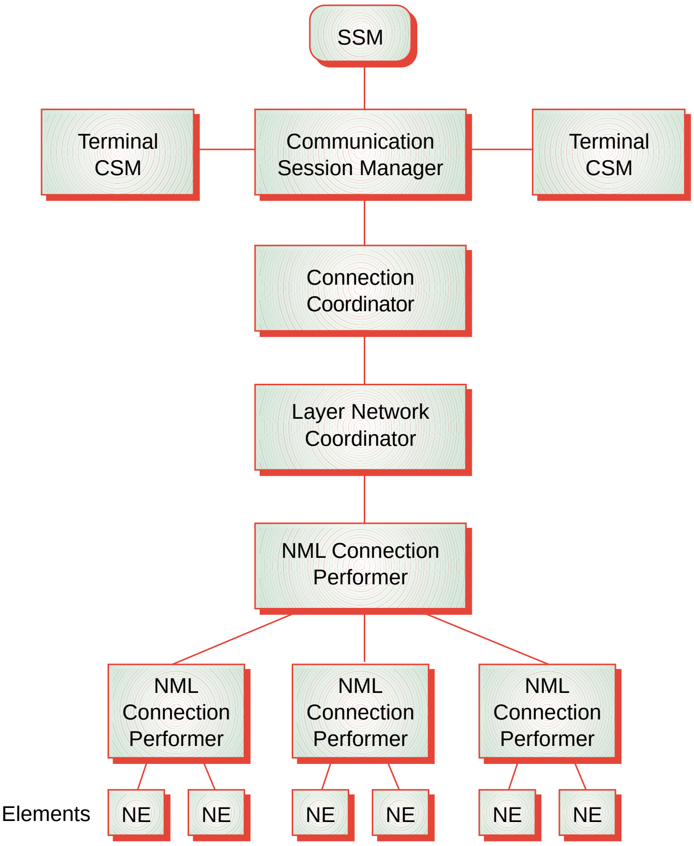


Figure 8 Overview of the network resource architecture

A flow termination point is an endpoint of a flow and may represent a hardware device such as a microphone or a loudspeaker, or software component. Flow termination points are located within terminals. Since terminals and networks may have different capabilities, an important part of flow establishment is to agree on the quality of service and coding formats, e.g. MPEG or H.261 for video, to use.

The central components within this layer are called the Communication Session Manager (CSM) and Terminal Communication Session Manager (TCSM). The CSM has the overall responsibility. It provides service components with a view of connections that is “abstract”, i.e. independent from specific network technologies. TCSMs run within terminals and represent the terminals for the purpose of establishment and management of flows. The CSM thus interacts with the TCSMs and the CC (described below) to establish the flows.

The abstract nature of the interface provided to services by the CSM is very important since it enables the construction of services that can run on different underlying network technologies.

Below the Communication Session we find the *Connection Session* layer. The central component within this layer is the Connection Coordinator (CC). The CC provides the CSM with interfaces to interconnect network access points. The CC hides underlying network technologies, and provides an abstract view of the connections. It is able to use different underlying network technologies to establish the connections. It will pick one or more specific underlying networks (“layer networks”) based on the Quality of Service requirements on the connections expressed by the CSM. Thus, the CC is the component of the architecture that maps the abstract, application oriented QoS requirements down to requirements on a specific network technology, e.g. traffic class, bit rate, end-to-end delay, and cell delay variation. The CC also handles inter-working between different network technologies.

The bottom layer of the NRA computational architecture is the Layer Network. The central component here is the Layer Network Coordinator (LNC). It deals with the set-up and management of connections within a specific type of network. The layer network may be structured as a hierarchy of subnetworks. Each subnetwork is controlled by a Connection Performer (CP) component.

Two noteworthy points concerning the NRA are:

- It has a strictly hierarchical structure and is an example of a class of connection control systems referred to as “open signalling systems”. The idea is to implement the network intelligence, i.e. routing algorithms and QoS management, in computers outside the switches. The switches themselves are simple and “dumb” and controlled through a simple, low level interface, see Figure 9.
- It provides a common architecture for control *and* management. This is in contrast to traditional network systems, where a distinction is made between signalling and management applications, the two being based on totally different software architectures.

3 Relationships to other work

This chapter contrasts TINA with related work within the areas of multimedia services, intelligent networks and broadband networks.

3.1 TINA and other approaches to multimedia services

Regarding full multi-media multi-party services, there are several existing standards in the area, like the ITU standards H.32x for conferencing over POTS, ISDN, LAN and IP networks. In addition, there has long been activities in ITU SG11 for ‘call control’ of multiparty services inside the framework of B-ISDN signalling. The purpose of this chapter is to contrast TINA to some of these approaches.

3.1.1 The H.32x family

The ITU standards H.32x focus little on conferences on demand. The conferences are planned in advance and typically ordered by fax or web. Also the location/terminal of the different participants are agreed *in advance*, so personal mobility is not an issue, and not explicitly covered.

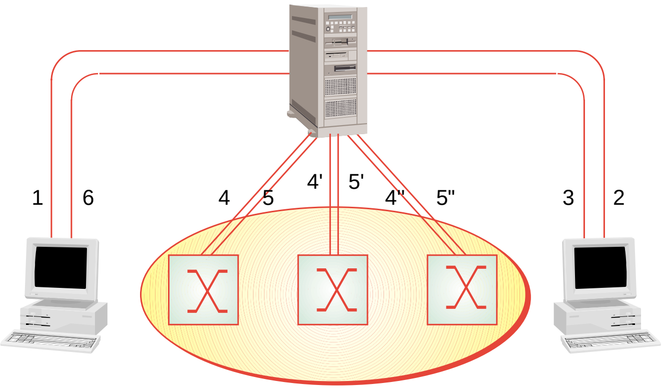


Figure 9 Open signalling

The H.32x family of systems does not encourage individual tailoring of services to individual users. By using generic operations, the number of interactions among the involved parties tend to be quite large. Normally, the service takes several or many seconds to establish. Payment, subscription, etc. are out of the scope and must be handled by proprietary means.

There are different versions of this standard for different types of networks. We consider H.320 for N-ISDN and H.323 for IP as most important and will cover them specifically.

3.1.1.1 H.320 over ISDN

H.320 standardises multimedia conferencing over ISDN. In H.320 the control operations are sent in-band. First a plain ISDN bearer channel (B-channel) is established from each of the conference participants to the service provider. The service provider negotiates the coding formats, QoS, number of B-channels, etc. with the conference participants over this channel. After this set-up phase both control and media streams are sent over the agreed number of B channels.

TINA, in contrast, sends control information on a logically separate network, which is a better approach when control information is exchanged before the media streams are set up. Since TINA, as opposed to H.320, supports personal mobility and ‘on-demand’ invitations, no particular assumptions regarding network technology are made at the service control level. Instead, the kTN is used to carry the control information.

Note, though, that from the network point of view, pure ISDN is used to support H.320 conferencing. All the conferencing related service control is done end-to-end between the different end users and the service provider, who is also an end user seen from the ISDN network’s point of view. The conferencing service unit (MCU) and the software for conference control can be owned and operated by a service provider different from the ISDN provider. Thus, they separate service provider and network provider and allow for business relationships quite like the way TINA does.

3.1.1.2 H.323 over IP

H.323 standardises multimedia conferencing in LANs and IP networks. H.323 share with TINA the principles of separation of “service control” from “connection control”. This is done through three different protocols: RAS, H.245 and H.225. H.323 does not, however, focus on conferencing across different administrative domains. The focus so far is on conferencing within a single domain, called an H.323 *zone*.

The *gatekeeper* is the unit providing access control and bandwidth inside one H.323 zone. Conferencing across several H.323 zones is currently for further study; this is linked to the development of IP reservation mechanism like RSVP and how zone or domain boarders will affect the reservation protocols in IP.

H.323 version of conferencing addresses many other areas as well, such as IP multicast, and gateways between H.323 zones and basic ISDN terminals and H.320 compliant terminals are defined. Thus, terminals outside this H.323 zone, placed in an ISDN network, can partici-

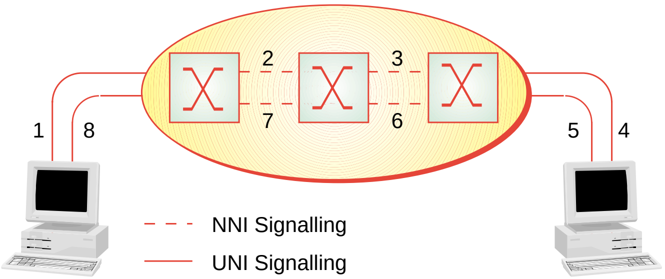


Figure 10 Classical signalling

pate in this conference service without involving any conference service at the ISDN side.

We may note that the H.323 standard for conferencing over IP has been developed quickly and mainly by software companies, and then put into ITU to be 'stamped'. New versions of the standard are under development.

3.1.2 Control of multimedia streams in OMG

The OMG has just adopted a standard called 'Control of A/V Streams'[1]. Products supporting this standard are expected some time during 1998. The OMG standard can be seen as a simplified version of the TINA concept of communication session. The proposal is specifically aimed at being well suited both for ATM and for IP networks. This illustrates how the TINA communication session concept is well suited to fit between TINA Service Architecture and different underlying network technologies. It also relates to H.245, being sort of a 'CORBA-fied' version of it.

3.1.3 B-ISDN Call Control

In B-ISDN signalling there is the concept of 'call control'. In addition, IN triggering mechanisms, like N-ISDN Q.1200-CS3, are assumed in cases where more advanced service session control is needed. This means that service control is split between the basic B-ISDN signalling and IN. Basic B-ISDN signalling contains generic service operations like join, ownership control, etc. in addition to pure connection establishment. The proposed mechanisms for IN-triggering are quite complex, so that e.g. personal mobility and other service related issues might be handled *during* connection establishment, instead of *prior* to connection establishment.

As the focus over the last few years has shifted towards multimedia over IP, it may be questioned whether service control for this type of services will ever be handled by the far less open B-ISDN signalling system. Basic connection control may, however, be handled by B-ISDN signalling.

3.2 TINA and IN

TINA has sometimes been referred to as 'the next generation IN'. There is some element of truth in this expression; there are similarities between TINA and IN. There are also, however, great differences, both in scope and technical approach. The similarities and differences will be discussed in the following. The considerations are related to scope, user interface, support for mobility, and support for rapid service development.

Both TINA and IN focus quite strongly on value added services, like freephone and premium rate. But unlike TINA and H.32x conferencing services, IN is targeted towards enhancing N-ISDN and plain telephony ('POTS') calls between two endpoints. Other services not based on N-ISDN, or not based on a two party call, are excluded from current IN. This involves most of the multiparty and/or multimedia services.

All IN services can be used from a plain telephone without any extra equipment. This usability from every phone is a major strength with IN. As GSM phones, ISDN phones, and palm tops become increasingly widespread and have increasingly good displays, this strength will, however, be less important in the future. Moreover, many services, e.g. ticket booking, are quite clumsy to use with the telephony interface and could obviously be handled better with a text/icon based user interface, more like the one on the web.

In IN, all outgoing calls are assumed to be placed from a telephone, and all messages to the involved parties are supposed to be delivered over voice as well. Future multimedia services violate this assumption, however. If a user wants to 'call' a 'multi-media UTP' number, he might as well start from a PC, and if the called party is not available, electronic communication, like e-mail, may take place instead. Under such circumstances the basic assumptions in IN are invalidated, and a voice connection never really needs to be established.

The assumption in standardised IN is that BCSM is always used. This translates to an assumption that every service invocation starts with the initiation of an N-ISDN voice call. This assumption allows personal mobility to be handled during the establishment of the voice connection, i.e. at the originating side for outgoing calls, and at the terminating side for incoming calls. In TINA, mobility is handled through interactions over the kTN (signalling network) before media connections are established. This approach seems to be better sited to support a multi-media and multi-network environment.

The most important goal of IN is to support rapid development of new services. IN has not completely lived up to its expectations in this area. This is partly because services are based on standardised modules called SIBs (service independent building blocks). Deployment of new SIBs in IN tends to require upgrades also in the switches. Another reason is that the BCSM and the IN-triggering tend to make quite complex interactions between INAP and ISUP signalling.
A third reason being that the software development environment in IN is a closed one, and that specialised skills are needed to do 'IN-programming'. This is opposed to other approaches towards ser-

[1] *The term* A/V stream *in one OMG corresponds to the more generic term* multi media stream.

vice provisioning, notably the Internet and TINA, where open interfaces and standard application development tools are used.

One may suspect that very little from existing SIBs and IN software may be reused in future multi-media, multi-party services. Rather, principles from H.32x and from the TINA service architecture may be highly relevant in such a context.

3.3 Signalling in B-ISDN and ATM

Traditional signalling techniques come from the telephony world and these paradigms are still prevalent in broadband networks. For example, the ATM standard for signalling is now Q.2931 which is based on the narrowband ISDN Q.931 standard. The basic principles of this signalling paradigm is illustrated in Figure 10.

There are several limitations to Q.2931. For this reason, there are currently enhancements under study within ITU-T and ATM Forum. The limitations can be summarised as:

- The model is geared to running a single connection within a single call. Much more flexibility is needed to support multimedia and conferencing services. There is a need for complex topologies, and some notion of a session to encompass these multi-party connections.
- The implementations of UNI signalling have too large a footprint. All code needed to run the protocol has to be built into the terminal. The switch also needs to run the entire protocol leading to a huge amount of code running on the switch. The code is also all mixed up with session level logic about what the end parties can do.
- More than the code, the specification of the ATM-Forum UNI requires almost 150 text book pages to define the connection management signalling. This is because it must define the delivery semantics and encoding at the bit level. An equivalent CORBA based interface for slightly more functionality requires dramatically fewer (10 or 20) pages and is mainly self documenting. The expression ‘CORBA based’ automatically defines the delivery semantics and encoding.

The TINA Network Resource Architecture is based on a totally different paradigm, often referred to as *open signalling*. The paradigm, illustrated in Figure 9, allows third party providers to develop hardware independent signalling systems. This software utilizes low-level open interfaces for switch control *and* configuration.

The potential benefits of this architecture are:

- The implementation is simpler, since it is based on an underlying DPE.
- Different, more centralised approaches to routing becomes feasible.
- A common architecture can be used for both control (e.g. signalling) and management.
- The architecture is well suited to support the complex connection topologies needed by multimedia and conferencing services

The major obstacle for the TINA NRA and other architectures based on the open signalling paradigm is the fact that it differs so radically from traditional signalling systems and lacks support from the major equipment vendors and standards bodies.

4 Future Work

The first phase of TINA-C ended in December 1997. In January 1998 the consortium embarked on a second phase with a modified organisation. The most important change is that the Core Team has been dissolved. According to the plan, the consortium will now be driven by Working Groups consisting of voluntary participants from the member companies. The consortium will be controlled by the Consortium Forum (CF) consisting of representatives from all member companies, and an Architecture Board (AB). The AB will consist of a limited number of representatives, and is responsible for the technical coherence of the work plan, and for setting up the Working Groups. With the new organisation, TINA-C also introduced multiple membership classes, the idea being to increase the number of member organisations, and especially to encourage universities to join.

The decision to organise an extension of TINA is clearly motivated by the fear that the architecture and specifications will have very little impact on the industry if the Consortium is closed now. Firstly, there is a need to promote the results to a wider audience than the relatively closed group of companies that have been involved so far. Part of the strategy is to try actively to involve more organisations in the consortium, and to spend resources on large scale demonstrations of services based on the TINA architecture. Several demonstrations are planned during 1998, involving major companies.

Moreover, there are several unfinished parts of the architecture. Several reference points have not been specified. Many feel that the architecture needs simplification, and strategies for migrating from existing telecommunications architectures to TINA are not adequately described.

At the time of writing, the following Working Groups (WG) have been established:

- DPE WG. The purpose of this WG is to continue the clarification of the additions required in the area of Distributed Processing Environment to the OMG specifications, and to continue the effort to encourage adoption of corresponding specifications by the OMG.
- Service Management WG. The purpose of this WG is to define a framework for service management, recognising that the service management area is poorly covered in other forums.
- IN and TINA WG. The purpose of this WG is to specify strategies to migrate from services based on ITU-T IN to TINA based services, and to promote and contribute elements of the TINA architecture to the groups within ITU-T standardising IN and UMTS.
- Next Generation Mobility WG. The purpose of this WG is to further develop the work done by TINA within the areas of personal mobility and terminal mobility, and to contribute the results to standardisation bodies working with future mobile networks, such as UMTS.

An unanswered question is whether the best approach is to pursue the TINA ideas within a TINA consortium, or whether to do it within other forums, such as the OMG Telecommunications Domain Task Force, which includes many of the same members, and even the same people.

5 Concluding remarks

The TINA architecture promotes many principles that apparently would have great benefits to different stakeholders in the telecommunications industry if they were adopted. By hiding the complexity of the networks from the services, the TINA architecture could help operators develop common services over different network technologies, and to introduce new services more rapidly, thereby reducing operations and maintenance costs. Furthermore, the use of a generic distributed processing environment ensures portability across multi-vendor equipment, the telecom industry to benefit from advances in computing technology, e.g. object orientation and distribution. Finally, the use of a common architecture for all kinds of services would enable manufacturers to adopt a common approach to services and their management.

Although many of the underlying principles of TINA are useful, it is not currently clear that TINA as such will be widely adopted. One of the major problems of the architecture is its huge scope. If introduced all at once, it would represent a revolution in the telecommunications industry. That is not likely to happen. Of course, TINA is not necessarily a question of all or nothing. A system may use only part of the TINA architecture, neglecting other parts. Clear strategies are lacking, however. A second problem is the lack of industry support outside the research organisations of the member companies. So far, TINA products are non-existent. Finally, TINA faces competition from other initiatives with partly overlapping goals. Some of these initiatives have a stronger industry support and move much faster than TINA, e.g. the OMG, the ITU-T H.32x series of recommendations, the Java community, and several activities within the Internet community related to multimedia services and future network infrastructure.

6 References

Note: The TINA documents are available from the TINA-C Web site http://www.tinac.com.

1 Darmois, E, Hoshi, M. *Software Architecture for Multimedia and Information Services.* Global Communications, 1997.

2 *Informational Modelling Concepts, Version 2.0.* TINA-C, April 1995.

3 *Computational Modelling Concepts, Version 3.2.* TINA-C, May 1996.

4 *Engineering Modelling Concepts.* TINA-C, December 1994.

5 *TINA Service Architecture, Version 5.0.* TINA-C, June 1997.

6 *TINA Network Resource Architecture, Version 3.0.* TINA-C, February 1997.

7 *TINA Business Model and Reference Points, Version 4.0.* TINA-C, May 1997.

8 *Ret Reference Point Specification, Version 1.0.* TINA-C, September 1997.

9 *The ConS Reference Point, Version 1.0.* TINA-C, November 1996.

10 *The Tcon Reference Point, Version 1.1.* TINA-C, November 1996.

11 Solbakken, H et al. CORBA as an Architecture for Distributed Systems. *Telektronikk,* 94 (1), 107–118, 1998 (this issue).

12 Rumbaugh, J et al. *Object-Oriented Modeling and Design.* Englewood Cliffs, NJ, Prentice Hall, 1991.

13 *Service component specifications part B, Version 1.0.* TINA-C, January 1998.

Tom Handegård is Research Scientist at Telenor R&D, Kjeller, where he has been employed since 1990. He has been working with Intelligent Networks, B-ISDN signalling, and the TINA architecture. Currently, his main interests are within various aspects of distributed object-oriented systems, including CORBA and Java technology.

e-mail: tom.handegard@fou.telenor.no

Lill Kristiansen holds a PhD (Dr. Scient) in mathematical logic from the University of Oslo in 1993. She worked with object oriented methods and tools applied to IN and TINA at Telenor R&D from 1993 to 1997. She is currently working as process and methodology coordinator at Ericsson, in the Internet/Broadband group. She is also engaged at University Studies at Kjeller (UNIK) as associate professor.

e-mail: etolkr@eto.ericsson.se

CORBA as an Infrastructure for Distributed Computing and Systems Integration

HÅKON SOLBAKKEN (EDITOR), OLE JØRGEN ANFINDSEN,
EIRIK DAHLE, TOM HANDEGÅRD, KJELL SÆTEN

This article provides an overview of the Common Object Request Broker Architecture (CORBA) as a technology for distributed computing and systems integration. CORBA is both robust and commercially available. It provides a relatively high-level solution for program-to-program communication, interoperability across different platforms, networks and languages, and a rich infrastructure for developing, integrating and running distributed applications.

There is a trend in enterprise application development towards three-tier or multitier architectures with thin Web-based clients and integration of legacy systems. CORBA fits well into this picture, and major computer software vendors have chosen CORBA as a key technology for the next generation Web.

CORBA has become an interesting technology for telecom applications as well. OMGs Telecom Domain Task Force is currently working on how to introduce CORBA in telecommunication management systems, including systems based on TMN and SNMP, and in Intelligent Networks (IN).

1 Introduction

The design and implementation of software is a difficult and expensive activity, especially when the software has to run on a network of machines, with the overall functionality distributed among the machines, and when the software has to be integrated with other systems, such as legacy systems or purchased systems. This is increasingly the situation today. To add to the complexities, today's software often has to deal with heterogeneous environments of different languages, platforms and network protocols.

The challenges of developing *distributed* software stem from fundamental problems, such as detecting and recovering from network and host failures, and from limitations with tools and techniques used to build such software. The challenges of *integrating* systems stem from the use of different models, languages, interfaces and protocols.

Distributed object computing is a promising approach to distributed computing and systems integration. It combines two major areas in software technology: distributed computing systems and object-oriented design and programming. Distributed computing system techniques focus on how to provide support for resource sharing, openness, concurrency, scalability, fault tolerance and transparency [1]. Object-oriented design and programming, on the other hand, focus on reducing complexity through modularization, encapsulation of data and methods, inheritance, polymorphism and reuse of existing components.

The two most widely adopted distributed object computing models are the Common Object Request Broker Architecture (CORBA), which is standardised by the Object Management Group (OMG), and the Distributed Component Object Model (DCOM), which is being developed by Microsoft.

CORBA defines how client applications can invoke operations on server objects, no matter where they are located and who designed them, using the services of an Object Request Broker (ORB). The OMG has two main goals with CORBA [2]. Firstly, to make it easier to implement software that must bridge the boundaries of different platforms, networks, and languages. Secondly, to encourage the development of open software that can be used as components of larger systems. The ultimate goal is to reduce complexity, lower cost and hasten the introduction of new software applications.

CORBA specifications and implementations have been evolving over several years, and they are still under construction. Although the first CORBA specification was defined in 1991, and CORBA products have been available since 1993, it is just in the last year or two that really workable standards as well as complete and robust implementations have been produced.

The adoption of CORBA technology has increased rapidly in the same period, partly because of the marriage of CORBA and Web-technology. Major computer software vendors such as Netscape, Oracle, Sun and IBM have chosen CORBA as a key technology for the next generation Web. The new Web, called the Object Web [3], will be used to build and deploy full-blown business applications based on three- and multitier architectures, and thin clients running in Web-browsers, providing a globally accessible user interface.

Over the last two years the authors of this article have both studied and gained practical experience with CORBA. As part of the ACTS ReTINA project, we are currently building a BVPN (Broadband Virtual Private Network) demonstrator using CORBA [4]. The main aim is to test the ReTINA DPE (Distributed Processing Environment), a real-time distributed processing environment based on CORBA.

The aim of this article is to provide an overview of CORBA and how to use CORBA to develop distributed applications. We provide an introduction to OMG and the CORBA standard and look into how to use CORBA to implement applications. We present an architecture for CORBA applications and explain how this architecture can be enriched by the use of Java and Web-technology. We also describe how CORBA can be integrated with database systems and Transaction Processing (TP) monitors, and how CORBA can be used for systems integration. After looking into these general issues about using CORBA, we present the status with respect to the use of CORBA in telecom applications. The last part of the article compares CORBA with DCOM and other distributed object technologies, and presents an overview of CORBA products and also the future directions for CORBA.

2 OMG and CORBA

OMG was founded in May 1989 as an international industry consortium consisting of eight companies: Sun Microsystems, Unisys Corporation, Data General, Canon Inc., Philips Telecommunication N.V, Hewlett-Packard, American Airlines, and 3Com Corporation. This list has now increased to over 700 members world-wide, and it is the largest software consortium in the world today. The organisation is a non-profit corporation. It does not develop software itself, but rather makes specifications for development and deployment of heterogeneous distributed systems.

In order to make a specification within a certain area (e.g. naming service), a task force within OMG sends out an RFP (Request For Proposal). Each OMG member that intends to respond to an RFP, whether individually or jointly with other members, must submit a letter of intent (LOI) to respond to OMG by a date specified in the RFP. This date is

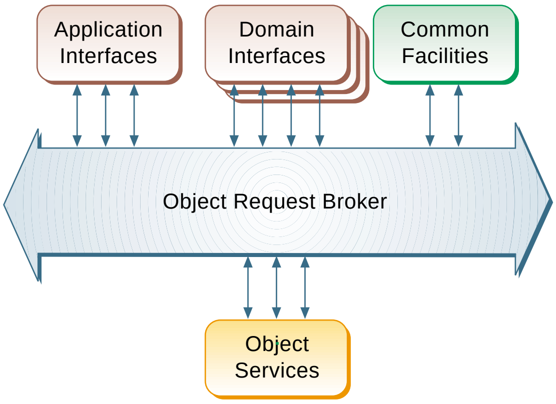


Figure 1 Object Management Architecture

typically 60 days before the initial submission date. Often more than one proposal will be received, and a review and voting process is conducted to select one of them. OMG will then adopt the specification and vendors can start the implementation. Note that after a member sends a proposal to an RFP, it must have an implementation within six months. Therefore, most of the adopted specifications are based on proven technology.

Sometimes a Request for Information (RFI) is sent out prior to an RFP. The RFI is a general request to the computer industry, universities, and any other interested parties to submit information about a particular technology area to one of the Task Forces in OMG. Based on the responses, an RFP might be sent out.

2.1 The Object Management Architecture

OMG has specified a high-level architecture called Object Management Architecture (OMA). OMA provides a framework of services, facilities, and interfaces for distributed systems, and it consists of five components.

- *Object Request Broker (ORB):* is the software communication bus that all objects use to communicate.
- *Object Services:* provides fundamental, commonly used functionality to other CORBA objects.
- *Common Facilities:* while Object Services provides services to objects, facilities provide functionality to applications. Example of such a facility is the System Management Facility.
- *Domain Interfaces:* these are interfaces for specific application fields or domains, like telecom, finance and health.
- *Application Interfaces:* are interfaces for a specific application, and not part of the OMG standard, since OMG does not develop applications, and probably never will.

The rest of this section will concentrate on the ORB architecture and the Object Services. While a lot of the Object Services have been specified, the domain interfaces and the common facilities are the most recent areas of focus, and will not be described in more details.

2.2 CORBA

The most important part of OMA is the CORBA specification. It describes in detail the interfaces and characteristics of an ORB. It contains all the necessary functionality to identify and locate objects, handle connections, invoke methods on objects, and pass data between objects. The latest revision of the CORBA specification is CORBA 2.0, and the major components in it are: ORB core, Interface Definition Language (IDL), Inter-ORB Protocol (IIOP), static and dynamic invocations, language mappings, and Object Adapters. We will start by explaining what IDL is, then give an example of how to use CORBA (using Static invocation), followed by a short description of dynamic invocation, interface repository, and object adapters.

2.2.1 IDL

In order for a client to communicate with a CORBA object it needs to know what operations or services that object can provide. This is often called the interface of an object and in CORBA this interface is specified using IDL. Even though IDL has a somewhat similar syntax to C/C++, it is not an imperative programming language, but a declarative language. This means that the interface and the implementation are separated, and the implementation can be programmed in any programming language with a mapping to IDL. Languages with a mapping from IDL include C, C++, Smalltalk, Cobol, and Java. A language mapping means that each element in IDL is translated to a construct in the target language using an IDL compiler. An interface in IDL is for example mapped to an interface in Java, and to a class in C++.

So what does IDL look like? Figure 2 is an example from an ATM configuration management system.

Modules are used to control name spaces. To refer to *LayerNetwork* above, you need to use the module-name: *NRCM::LayerNetwork*. *LayerNetwork* is an interface, inheriting methods and attributes from *ATMObject* (not shown). Interface inheritance is an important feature in IDL, and it allows reuse of existing interfaces. All the methods and attributes in *ATMObject* will be inherited into *LayerNetwork*. *LayerNetwork* adds two methods, *createSubnetwork* and *updateBandwidth*. The *createSubnetwork*

```
module NRCM {
  interface LayerNetwork : ATMObject {
    exception AlreadyExists { };
    Subnetwork createSubnetwork(in string name)
    raises (AlreadyExists);
    void updateBandwidth(in string name, inout long bw);
  };
};
```

Figure 2 IDL example

method creates a new *Subnetwork* object with the name given as *in* parameter *name*, and it returns an object reference to the newly created *Subnetwork* object. *updateBandwidth* takes a layer-network name and a bandwidth (*bw*) as *in* parameter. *bw* is also an *out* parameter, and it returns the available bandwidth of the layer-network. IDL operations can raise exceptions to indicate the occurrence of an error. The *createSubnetwork* method checks if a subnetwork with the same name already exists, and if it does, it gives an *AlreadyExists* exception back to the calling object.

IDL is a simple, declarative interface definition language. It does not include features for defining object behaviour. It allows mappings to both object-oriented and non-object-oriented languages. It is possible to have a mapping to for example COBOL, which is important when encapsulating legacy systems.

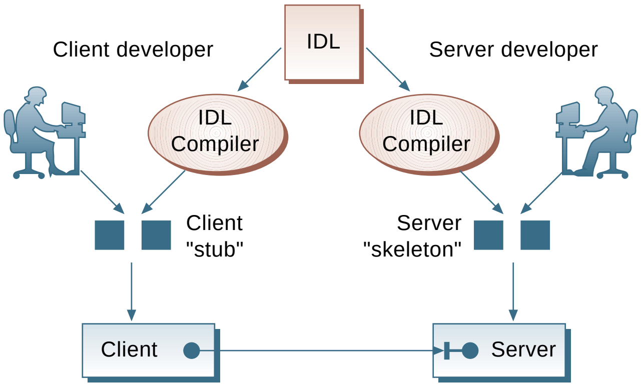


Figure 3 The CORBA development process

2.2.2 CORBA development process

Now that you know IDL, let us see how you can write a client/server application.

Figure 3 shows the development process. You start by writing the IDL in a file, and the interfaces will be the contract between the client and the server program. A client programmer runs the IDL file through an IDL compiler to produce stub code in the selected language. He then writes a client program which invokes the operations defined in the IDL file, and then link this code with the stubs file. The client is then ready to go. A server programmer will start his work by running the same IDL file through an IDL compiler to produce skeleton code in his favorite language. He then implements the interfaces in the IDL file, and compiles and links them with the skeleton code. Note that client and server programmers can select different implementation languages. So what happens when the two programs are executed? The server program creates some CORBA object, and the client gets a reference to this object. It can get this initial reference either from a Naming Service, from a proprietary bind mechanism, or by importing a stringified object reference from a file. This object reference is really a pointer to a local proxy that represents the object on the server. The client does not need to know where the real object is or anything else about it, except that it must provide the functionality specified by the IDL. The client can be written in Java, running on Windows, while the server can be written in C++, running on Solaris. When the client calls a method, it really calls the local proxy, which sends a request to the server, which calls the skeleton code, which again calls the real implementation. The implementation code is executed, and then the result is returned in the opposite direction. The strength of this approach is that the environment in which the client and the server execute can be completely different, they are still able to communicate with each other.

2.2.3 IIOP

CORBA 1.2, which came in 1991, did not specify the protocol it would use for sending requests and data between objects. This led to a different, proprietary protocol for each CORBA vendor. In 1995, the CORBA 2.0 specification was released, and one of the most important features was that it specified an on-the-wire protocol for interoperability between ORBs from different vendors. This protocol is called Internet InterOrb Protocol (IIOP), and it is running over TCP/IP. An IIOP packet consists of all the information an ORB needs, the target object, the operation being called, and the parameters in the call. IIOP provides the protocol, but something more is also needed in order to provide interoperability between different ORBs. The object references that each ORB uses must be standardised. CORBA 2.0 does this by specifying an Interoperable Object Reference (IOR). This means that if you have an IOR to an object, you can talk to it. It contains information about the machine-name and the port of the server, the name and the type of the called object and so on.

2.2.4 Dynamic Invocation and Interface Repository

Static invocation is appropriate for producing client/server application software. However, sometimes you want to be able to talk to an object without knowing its interface at development time. This is the case for tool-builders who want to make tools that list methods and attributes and possibly invoke methods on any object. OMG has called this dynamic invocation and it is provided in CORBA 2.0 by using what is called Interface Repository (IR). Whenever you compile an IDL file, you can specify that you want the metadata to be inserted in the IR. A client that wants to use the object without having the IDL file, can then ask the IR for the metadata and construct a call to the object at runtime. The drawback with dynamic invocation is that it is cumbersome to program and fortunately most client/server application programs do not have to use it.

2.3 CORBA Services

The idea behind the CORBA Services is to provide fundamental services to application objects. OMG has defined the interfaces (in IDL) and the semantics of each of the services. The implementations of the services are not standardized, and it is up to each vendor to make sure they deliver the functionality specified by OMG.

There are 17 object services as per today, and more might be added later. The 17 services are: Collection, Properties, Concurrency control, Query, Event notification, Relationships, Externalization, Initialization, Security, Licensing, Startup, Life cycle, Time, Naming, Trader, Persistence, and Transactions. Below is a description of some of the most important services.

Naming: This service is often compared to the white pages in a telephone directory. It allows a client to find an object based on a name. A server can register various objects by giving them a name, and the client asks the Naming Service for an object reference by specifying a name.

Event: This service is used for sending asynchronous events between objects. An object can ask the event service to be notified when a specific event occurs, like an attribute changing value in another object. The sender and the receiver of the event do not have to know about each other, and it allows one-to-many communication.

Trader: This service can be compared to the yellow pages in a telephone directory where it is possible to find objects based on some search criteria. It can also be compared to a Web-site search engine that returns object references instead of URLs.

Security: In all systems and especially in distributed systems, security is an all-important issue. CORBA addresses these issues in the Security Service. It is concerned with confidentiality and integrity of the data, accountability of users and availability of a system. Unlike some of the services mentioned above, security is an integrated part of the system and not just a standalone service.

Transaction Service: See section 3.4.

3 Using CORBA

In this section we will look into how to use CORBA in the realisation of applications.

3.1 Three-tier architecture

Traditionally, client-server applications have mostly been based on *two-tier architectures*; either with all the application logic on the desktop, the server running only the DBMS, or with a slightly thinner client and some application logic on the server, e.g. in the form of DBMS stored procedures. The main advantages of the two-tier architecture is its simplicity; it is quite straightforward to decide which functions go where, and normally all development tools can be purchased from one vendor.

Two-tier architectures have a number of serious shortcomings, leading to increasing popularity for *three-tier architectures*. These architectures are defined as follows. The first tier contains the clients, typically providing a user interface and local processing as appropriate. Generally, the clients can be kept small. The second tier (or ‘middle tier’) contains the application logic, while the third tier contains DBMSs and other back-end systems. The clients should never access the systems in the third tier directly. The middle tier makes abstractions of the resources in the third tier, and thereby shields the clients from the plethora of back-end systems. A replacement of a back-end system such as a DBMS with another one should not affect clients.

The advantages of three- over two-tier architectures include

- thin clients (only presentation logic). Thin clients are suitable for execution within Web browsers, which are rapidly becoming very popular as general front-end tools
- less communication overhead, since most of the data processing is done within the middle tier, located on the server
- DBMS independence, and ability to utilize data from multiple sources
- better support for inter-application communication and integration with legacy systems (see below).

The main drawbacks of three-tier architectures compared to two-tier architectures are the increased complexity and the fact that very often no single vendor provides the complete infrastructure.

CORBA applications are typically based on the three-tier architecture, as illustrated in Figure 4. Typically, clients provide the user interface and interact with middle tier objects over an ORB. The middle tier infrastructure may also offer the clients an interface based on Microsoft’s DCOM. In some cases this may be an attractive alternative for clients running on Windows platforms. OMG has developed a DCOM-CORBA interoperability standard, and gateway products are available from CORBA vendors.

The middle tier contains the objects, often called ‘business objects’, that implement the application logic. Business objects within this tier may themselves have client-server relationships. The term *multi-tier architecture* is sometimes used to emphasize this aspect of the middle tier.

The business objects can also utilize functionality provided by CORBA services, such as transactions and security services. They communicate among themselves and with the object services over an ORB. This means that the business objects can be distributed over a number of processing nodes. The partitioning of objects onto processing nodes depends on requirements on performance and the number of simultaneous clients. Additionally, object TP monitors can be used to improve the performance and fault-tolerance of the middle tier applications by providing mechanisms such as load balancing and object replication (see section 3.4).

The back-end tier contains databases and legacy applications. The middle tier objects communicate with the databases and applications in the third tier using whatever protocol or middleware available, CORBA compliant or not. CORBA vendors already support a range of adapters and mechanisms enabling CORBA objects in the middle tier to communicate with some of the most popular DBMSs and other legacy systems, see sections 3.3, 3.4, and 3.5 for more details.

3.2 CORBA, Java, and the Web

Java is an object-oriented programming language that has gained much attention over the last couple of years. The main reason for Java’s popularity is the language’s inherent support for portability, mobile code, Web integration, and a relatively rich, standard class library.

As with all popular programming languages, OMG and the CORBA vendors

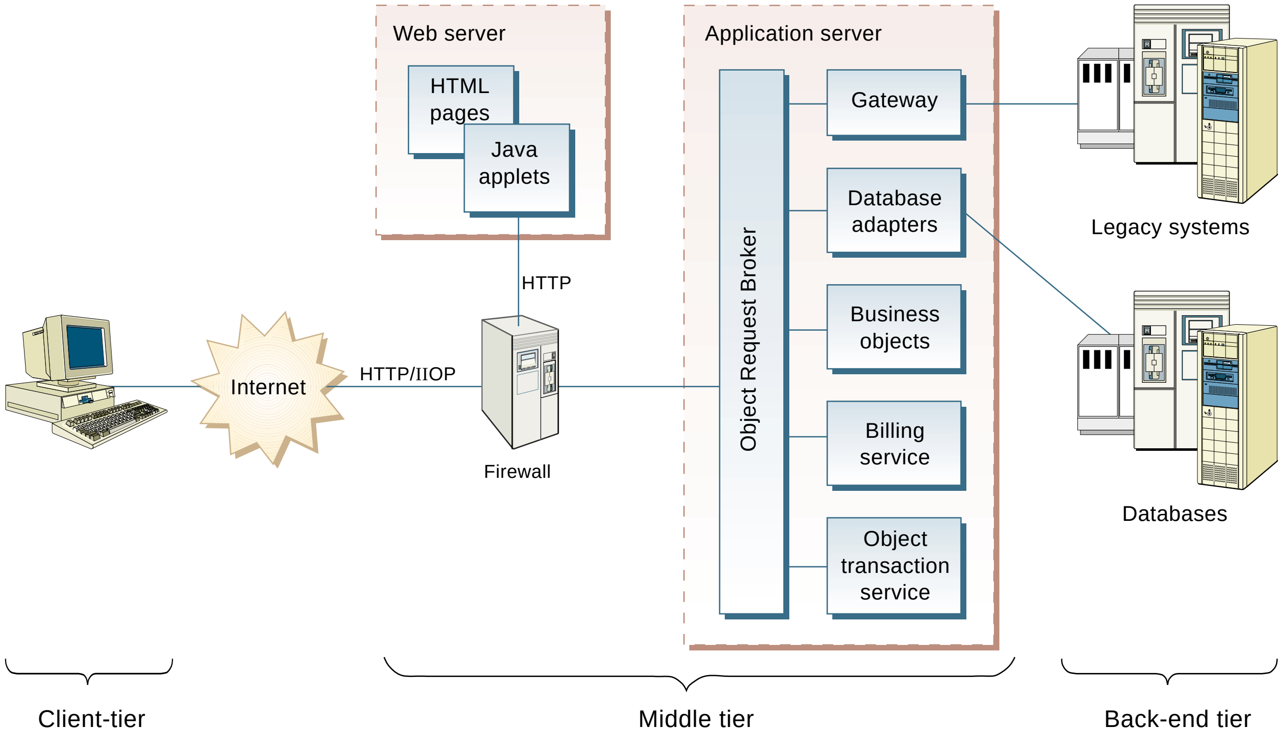


Figure 4 Three-tier architecture based on CORBA and Internet technology. The first tier contains the client components that provide a user interface and local processing as appropriate. Typically, the clients can be implemented as CORBA-enabled applets running inside Web browsers. The middle tier contains HTML pages and Java applets that are stored on Web servers, and objects that represent persistent data and implement business logic and are running on application servers. All the middle tier objects are interconnected by an ORB. The objects can utilize functionality provided by general services, such as transaction services, billing services, and security services. The back-end tier contains databases and legacy systems. They are integrated with the middle tier business objects through database adapters and gateways, respectively

have undertaken a number of efforts to provide support for the use of Java in CORBA environments. The reason for the strong interest in Java is that Java has a number of properties that are complementary to the properties provided by the CORBA architecture as such.

Firstly, Java provides the ability to write portable clients and object implementations. Once a CORBA object has been implemented with Java, it can be run anywhere without the need for recompilation.

Secondly, Java offers simplified code distribution. Client applications do not have to be installed on client computers, but are downloaded from a central (Web) server as needed. Note that in this case, there are two ways to support CORBA on the client (browser) side. The first alternative is to include a Java ORB within the browser. This approach is supported by Netscape, who provides a Java ORB with every instance of their browser. The second alternative is to download all the necessary ORB functionality from the Web server along with the applet (in this case often referred to as an ORBlet). The latter alternative has the disadvantage of increased download time on startup, but the advantage that it requires no pre-installed CORBA support on the client side.

Thirdly, Java supports Web integration. Client applications (Java applets) can be discovered through Web surfing, downloaded into Web browsers and be integrated within HTML pages.

Finally, Java is a simpler language to write object implementations than C++, the most common language used for CORBA objects so far. On the downside, the performance of Java is poorer than the performance of C++, although the arrival of JIT (Just-In-Time) compilers for Java has recently improved Java performance.

Through Java, CORBA is integrated with Web technology. The Java-CORBA combination is currently regarded by many (e.g. by Netscape and Oracle) as the most important element of an infrastructure for the next generation WWW, the so-called ‘Object Web’.

Note that in the Java case, the OMG is for the first time working on an *inverse* language mapping; a Java to IDL language mapping. The motivation is to enable automatic generation of IDL from Java interface definitions. This will in effect hide IDL from developers working with pure Java applications. Proprietary products supporting automatic Java-to-IDL are already available. These tools provide programmers with an environment very similar to the one provided by Java RMI (see section 5.2), yet at the same time preserve the interoperability properties of CORBA, e.g. allowing access to the Java objects from clients written in other languages, through the automatically generated IDL interface.

3.3 Integration with databases

Many CORBA applications will need support for object persistence. An object is said to be persistent if it is stored in such a way that it remains available for any application that needs it, even if the server process that manages the object is no longer active. Typically, persistent objects are stored in databases. If good support for object persistence is to be found in a CORBA environment, certain requirements must be met both by ORBs and DBMSs (database management systems).

The most important requirement for DBMSs is support for storing objects *as objects*, i.e. without having to disassemble the objects, e.g. into relational tuples, in order to store them. It is of course possible to make objects persistent by means of a DBMS that does not understand objects, but it will be cumbersome and/or yield poor performance. Products exist that map objects to relations, but storing object directly in databases appears to be a better solution. Two kinds of DBMSs are competing in this market: object-relational (ORDBMSs) and object-oriented (OODBMSs). The first group consists of relational database products with object extensions added to them, more or less according to the emerging SQL3 standard. The three products currently leading this trend are Oracle, DB2, and Informix, with Sybase and SQL Server as possible future contenders. The other group consists of DBMSs that have been designed for objects from the very beginning. The pertinent standard for this group is ODMG-93, from the Object Data Management Group (www.odmg.org). Some of the more well-known products are GemStone Objectivity/DB, ObjectStore, O2, Poet UniSQL and Versant.

The two most important requirements for ORBs are support for loaders and user-defined object names, also known as markers or reference data. A loader is a routine that is capable of loading a given object from a file or database in the event that a client tries to invoke a method on an object that happens not to be in memory. Without a loader, such a situation will result in an object fault, and the client's request cannot be processed. Support for user-defined object names is necessary because every time a loader brings a given object in from the database, it must be able to assign the same name to it. This is a prerequisite, because the object name is the key used by the ORB to determine whether or not an invoked object is in memory, and if not, the object name is used by the loader to find the requested object in the database. If the object name were to keep changing all the time, this way of doing things simply would not work. Neither loaders nor user-defined object names are required by CORBA, but we think that sooner or later OMG will have to specify a standard solution to these issues. The overall architecture for a CORBA server whose objects are stored in a database, is shown in Figure 5.

The glue that is used to tie together an ORB and a DBMS, is frequently referred to as a database adapter. In addition to a loader, a database adapter must have a module that can take care of memory management. This will typically involve things like keeping track of which objects are being referenced by clients, and preventing the server's memory from being overloaded with database objects (the number of objects in the database could easily exceed the capacity of the server) by choosing suitable victims whenever necessary (e.g. based on a least recently used algorithm) and throwing them out. Should such a victim be invoked by a client holding a reference to it, then the loader will automatically bring the object in question back into the server's memory.

Two CORBA services are particularly relevant in a database context, namely the Persistent Object Service (POS) and Object Query Service (OQS). POS provides generic interfaces that can be used with a variety of data stores, thus making it easy to port CORBA applications from one data store to another. However, it is now recognised that the first version of POS has been a failure, and OMG is currently working on a major revision. OQS enables clients to submit ad-hoc queries to the CORBA server, using a query language supported by the underlying data store, typically Structured Query Language (SQL) or Object Query Language (OQL).

3.4 ORBs and Transactions

If a server is to provide advanced services and/or support a large number of concurrent clients, one soon discovers that just having a CORBA-compliant ORB is insufficient. Developers of such systems will require more sophisticated environments, with support for transactions, concurrency control, naming, notification, security, system management, load balancing, and more. As mentioned elsewhere in this article, OMG has defined a number of CORBA Services that address these needs. However, there is also a need to integrate such services in a coherent way, and products that are capable of such integration are often referred to as TP monitors.

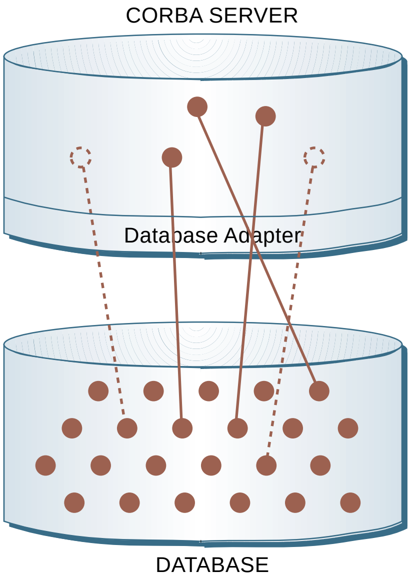


Figure 5 A database can contain a large number of persistently stored objects, and it may not be feasible to have all of those objects inside the server's address space at any one point in time. The solution is to have a database adapter that can throw out objects whenever that is necessary in order to make room for new ones. If a thrown out object (shown as a dotted circle in the figure) is re-invoked by a client, the database adapter, by means of its loader component, will bring that object back in from the database. This happens transparently to the client. Thus, any object in the server's address space has a persistent version of itself in the database, but the converse is not true

The most important benefit that TP monitors has to offer CORBA environments, is probably that of making client-server interactions transactional. In particular, this means that a client can make

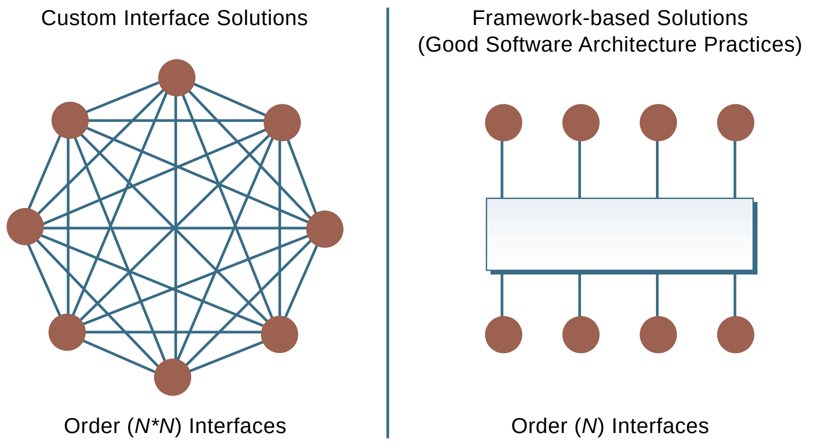

Custom Interface Solutions
Framework-based Solutions
(Good Software Architecture Practices)
Order (*N**N) Interfaces
Order (*N*) Interfaces

Figure 6 This figure illustrates one advantage of a framework-based approach to interoperability. If a gateway is to be built for each pair of systems that need to interact, the number of gateways needed will be proportional to the square of the number of systems. When using a framework like e.g. CORBA, one interface per system will suffice. (Figure adopted from [5])

several invocations on the server, and then decide whether to commit or abort the work thus performed. In other words, several operations can be grouped into a single logical unit of work (LUW), and either all effects of that LUW are applied to the server, or they are all wiped out, meaning the server will be left in a state *as if* the LUW in question never executed at all. This property is known as atomicity. If more than one data store is involved in a LUW, then one must in general use a protocol known as two-phase commit (2PC) to ensure atomicity. Any decent TP monitor will support 2PC, and therefore free CORBA programmers from the non-trivial task of implementing that protocol.

Two of the CORBA Services are of particular interest in this context, viz. the Object Transaction Service (OTS) and the Concurrency Control Service (CCS). OTS and CCS correspond to the two problem domains dealt with in transaction management respectively; (1) recovery – making sure that work done by transactions are either properly committed or rolled back, and (2) concurrency control – making sure that different transactions do not unduly interfere with each other. Both these aspects of transactions must be dealt with in order to ensure the classical ACID properties (atomicity, consistency, isolation, durability). Products that implement OTS and CCS are now available in the marketplace. The current trend seems to be to market OTS and CCS as key components of TP monitors, rather than selling them as separate products.

TP monitors have been successfully used in mainframe environments for some 20 years (e.g. CICS), and are just now beginning to emerge in the CORBA marketplace. Several TP monitors or OTS/CCS implementations for CORBA environments were available, or about to become available, by late 1997. Iona has teamed up with Groupe Bull and Transarc to offer OrbTP and OrbixOTS/OrbixOTM, respectively. IBM's Component Broker includes a TP monitor which, like OrbixOTM, is based on Encina from Transarc. And BEA Systems offers a Tuxedo-based TP monitor called ObjectWare.

3.5 Systems Integration

Due to its inherent ability to overcome problems related to distribution as well as language and platform heterogeneity, CORBA is positioned to play a significant role in the area of systems integration. In particular, many companies that use mainframes will find it interesting that by means of IDL their legacy systems can be accessed by clients running on PCs or Unix workstations. For example, an ORB running on a mainframe can provide access to applications running in CICS or IMS regions. This can be achieved in at least two different ways. One alternative is to have a gateway server that accepts an incoming CORBA method and then either builds a CICS Comarea which is passed to the CICS program or builds an IMS message which is submitted to the IMS input queue. The other alternative is to re-write existing applications so that they can act as CORBA server implementation objects. The latter provides more flexibility but requires an ORB that supports the native language of the legacy application in question, typically Cobol. For companies that have invested heavily in mainframe applications, integration with PCs and workstations could be a major incentive to adopt CORBA technology.

As a more general observation, one could say that OMG in its approach to systems integration has chosen a level of abstraction that is a lot more reasonable than some of the alternatives. Enabling integration by insisting that all systems be run on the same operating system, and/or use the same programming language, is unrealistic in the first place. Moreover, it would at best be a partial solution to the integration problems. At least two alternatives remain. One is to build a gateway for each pair of systems that need to interact, the other is to create a framework to which any system that obeys certain rules can be connected. The advantages of the latter approach over the former is illustrated in Figure 6. CORBA is an example of a framework-based approach to interoperability. It is interesting to observe that object-orientation, which separates external (object) interfaces from internal implementation details, seems to be an ideal foundation for interoperability frameworks. In the foreseeable future therefore, we expect any CORBA competitors to be other object-oriented middleware architectures, most notably DCOM from Microsoft.

4 CORBA for telecom applications

CORBA is rapidly becoming one of the technologies of choice for developing distributed systems in all areas of information technology including telecommunications. The following sections will discuss CORBA for telecom management systems, CORBA for IN, and CORBA support for multimedia applications.

4.1 CORBA and Telecommunications Management Systems

Telecommunications Management Systems in this context comprise Telecommunications Management Network (TMN) -compliant systems and SNMP-based systems. TMN is an architecture specified by ITU-T and ETSI, and applied by Network Management Forum, for management of telecom networks and services. SNMP (Simple Network Management Protocol) from IETF is used for management of data equipment, routers, etc. Both TMN and SNMP have a manager-agent communication architecture. TMN has selected OSI 'Guidelines for the Definition of Managed Objects' (GDMO) and 'Abstract Syntax Notation 1' (ASN.1) as the interface specification language, and OSI Common Management Information Protocol (CMIP) as the application layer protocol.

The JIDM group of Network Management Forum (NMF) in conjunction with the Open Group has pioneered the mapping between GDMO, CORBA and SNMP. They have a complete specification for a static translation between GDMO and IDL, and an interaction translation specification is underway.

The OMG Telecom Domain Task Force is looking at two aspects:

1. CORBA and TMN/SNMP coexistence and interoperability (a similar objective as NMF)
2. CORBA architecture inspired by TMN.

As regards interoperability, OMG issued an RFP titled "Interoperability between CORBA and Telecommunications Management Systems" in September 1997. Adoption of the specification is expected at the end of 1998. The scope of the RFP is to address the integration of GDMO/CMIP and SNMP management applications and protocols into a CORBA environment through an application protocol gateway. Only the GDMO/CMIP integration is covered here, but a similar approach is also requested for SNMP.

The integration will provide interoperability between TMN systems and CORBA systems, and preserve the investment in GDMO/ASN.1 information models. Proposals should address:

- Translation of GDMO to and from IDL specifications. Algorithms for static translation is requested, where static means that a GDMO to IDL compiler is used to generate static stubs/skeletons.
- Interaction translation. CMIP protocol data units are dynamically translated to one or more requests or replies on IDL interfaces.

Interaction translation is carried out by a gateway. The positioning of the gateway is depicted in Figure 7 and Figure 8. The specification translation is not shown in the figures, it takes place at compile time, and the result is used to implement the gateway.

As regards creating a CORBA architecture inspired by TMN, the most relevant specification is the Notification Service, which extends the CORBA capabilities of the CORBA event service to support filtering. A Notification Service RFP was issued in January 1997, and adoption of the specification is expected by March 1998. Event consumers can express interest in events that are filtered by type and content, and thus receive only a specific subset of all the events supplied to the notification service. The service provides a decoupling of consumers and producers, and shall satisfy the scalability demands (high volumes) of event-driven applications running within large networks including telecommunication management applications. Functionality requirements include dynamic addition of user-defined event types by producers, quality of service with respect to event delivery (guaranteed, priorities, best effort), federation of notification servers, etc. Also, there should be a specification of event types and contents applicable to particular vertical domains such as telecom (ref. TMN "X.700 series").

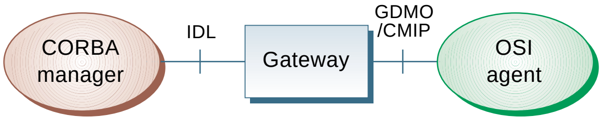


Figure 7 The gateway assumes the role of an OSI manager. This allows a CORBA system to manage TMN-compliant network elements and mediation devices, and is a likely way of introducing CORBA into the TMN architecture. The installed base of network elements remains as they are (provided that the network elements support GDMO/CMIP communication)

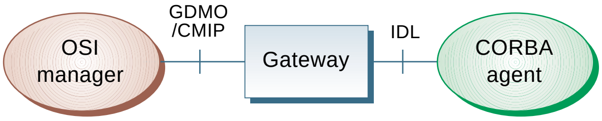


Figure 8 The gateway assumes the role of an OSI agent. This allows a TMN system to manage network elements with IDL interfaces, which may be interesting in the future. More important, the gateway's ability to play the role of both agent and manager means that CORBA management systems can interoperate with TMN management systems – in TMN terms referred to as peer communication between Operations Systems

4.2 CORBA and ITU-T Intelligent Networks (IN)

IN allows centralised processing of service logic and service data separated from the switching function in the switches, as opposed to traditional service implementations where a switch handles both the service processing and the switching. IN does not require that all switches are equipped with all services, and prevents heavy software upgrades in the switches. IN only requires the switches to support a service switching point (SSP), which during call processing is able to encounter an IN service triggering condition, and call a service control point (SCP) in order to process service logic and data. ITU and ETSI have specified both the architecture (called the IN architecture) and specific services (called IN services) following this approach. This text covers IN capability set 1, which is restricted to narrow-band telephony services. IN objectives include rapid service creation, tailored services, new services which rely on centralised data, well-managed services and cost-effective services.

IN has been implemented in the networks of telecom operators over the last years, but has only partly met the objectives. Existing IN implementations often suffer from being proprietary solutions. SCPs are usually closed environments which are inflexible with respect to integration with other systems. Service creation environments are special purpose and not based on state of the art software development tools. Specialised management systems are needed. Implementations are expensive.

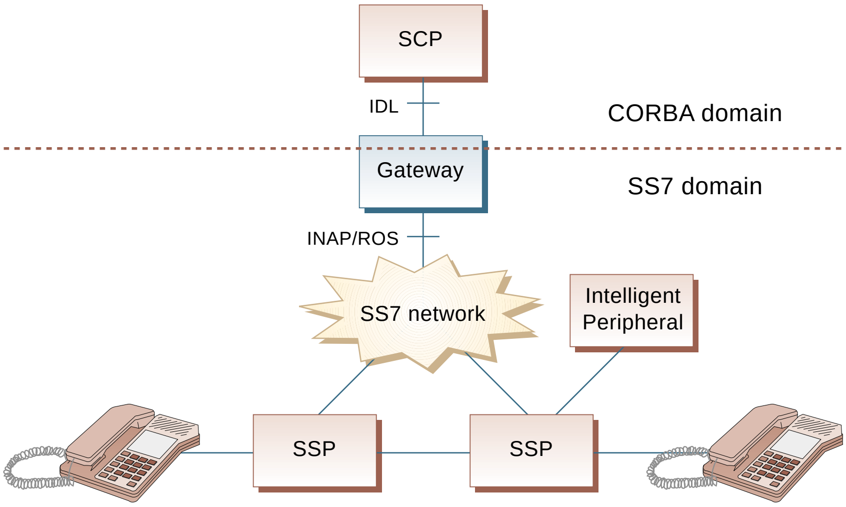


Figure 9 IN-CORBA gateway. The gateway allows development of centralised service logic and data (residing in the SCP) based on CORBA technology

The most realistic way of introducing CORBA into IN is to apply CORBA to the parts of IN which are separated from the switches. ITU and ETSI have standardised the communication between the SSP and the SCP, and the protocol is called INAP (Intelligent Network Application Protocol). INAP uses Transaction Capability Application Protocol (TCAP), which again uses the ISO Remote Operation Service Element (ROSE) protocol. ROSE implements the Remote Operation Service (ROS). INAP, TCAP and ROSE communication is carried by the signalling system No. 7 (SS7) network. A gateway between INAP/ROS and IDL operations would allow CORBA-based implementations of SCP and other ‘centralised’ IN entities including the Service Data Point and Service Management System. The prerequisite is that the network elements have to support standard INAP/ROS, which is not always the case.

It is expected that OMG will approve an RFP on interoperability between CORBA and IN by December 1997. The RFP will request an IN/CORBA gateway for interaction translation between INAP/ROSE messages and operations on IDL interfaces. There is a debate about the best protocol level for the gateway, i.e. whether there should be a ROSE to CORBA or INAP to CORBA gateway. The ROSE gateway would be very general (because ROSE is used by several application layer protocols including INAP and CMIP) and more complex, and could suffer from performance shortcomings. The most likely approach will be an INAP/CORBA gateway.

IN services have stringent requirements to performance, scalability and availability. CORBA technology will have to prove that these requirements can be fulfilled.

4.3 Control and Management of Audio/Video Streams

A specification for control and management of audio/video streams (hereafter called the AV streams specification) was adopted by OMG in September 1997. The AV streams specification introduces a wide range of application uses, including:

- Conferencing, entertainment, biomedical or security applications
- Distributed simulation or games
- High-bandwidth instrumentation in process or real-time control applications
- Bulk transfer of data, e.g. medical records, still images, software updates.

The specification covers topologies of streams, multiple flows, stream identification, stream set-up, modification and release, quality of service and relationship to network protocols. A stream represents continuous media transfer between end-points. Stream topologies covered are point-to-point and point-to multipoint. Three types of transport are supported by the framework, namely connection-oriented (like TCP), datagram-oriented (like UDP) and unreliable connection-oriented (like ATM AAL5). In principle, the specification allows a stream to be supported by any transport protocol including IP or ATM, hence the AV streams specification can be seen as an abstract framework residing on top of the transport. The AV streams specification has been influenced by the Telecommunication Information Networking Architecture Consortium (TINA-C) architecture, and the functionality is similar to the TINA Communication Session Manager component.

5 Alternative Technologies

CORBA is not the only middleware that support distributed objects. The two most serious competitors are Microsoft’s DCOM, and JavaSoft’s Java RMI.

5.1 Microsoft DCOM

DCOM (Distributed Component Object Model) is Microsoft's distributed object middleware and the most serious competitor to CORBA. The first version of DCOM was shipped as part of the Windows NT 4.0 operating system in mid-1996. DCOM is a distributed extension of COM, which was introduced in 1993, as part of OLE (Object Linking and Embedding).

DCOM has many of the same features as CORBA: An interface definition language (similar to OMG IDL), support for multiple implementation languages, static operation invocations, and dynamic invocations (similar to CORBA dynamic invocation interface, DII). Moreover, basic object services are available, notably naming, transactions and security. The object services are available as separate products.

The biggest difference between CORBA and DCOM lies in the object model. CORBA uses a 'classical' object model, similar to what is found in most object-oriented programming languages. DCOM objects on the other hand do not support multiple inheritance on interfaces, but instead provide reuse through more unusual mechanisms based on containment and aggregation. DCOM also supports a component model, based on so-called *AxtiveX controls*. An Active X control is simply a DCOM object that follows certain rules in its communication with other objects.

The heavy backing (i.e. Microsoft) and the technical features of DCOM make it a serious alternative to CORBA. The most important drawbacks are its complexity and the fact that it is only available on Windows platforms.

5.2 Java RMI

Java RMI (Remote Method Invocation) is a standard component of the Java system (JDK 1.1), available in every Java installation. RMI enables (remote) method invocations across Java Virtual Machines. The RMI package can in many ways be compared to an ORB as described in section 2. It enables clients to invoke methods on remote objects in a way that is similar to invoking methods on local objects.

Java RMI shares the basic principles with CORBA and DCOM. Clients interact with servers via interfaces. A local object (RMI stub) acts as a local proxy for the remote object. As in CORBA, the stubs are automatically created from interface specifications by a stub compiler. Moreover, clients locate and bind to server objects via a naming service (called 'repository').

RMI interfaces are, however, not specified using an external language (like CORBA IDL or DCOM IDL). Instead, native Java notation is used as the interface specification language.

Compared to CORBA and DCOM, RMI adds several innovations, including the ability to pass object values (state), not only object references, as parameters of remote invocations, dynamic downloading of object implementation code, and distributed garbage collection (although currently not bullet-proof).

On the negative side, RMI has some serious disadvantages. Firstly, RMI supports only Java-to-Java communication. It lacks a language neutral interoperability protocol. Moreover, the performance of RMI compared to Java ORBs is very poor. Also, RMI lacks protocol level support for security contexts and transaction contexts, lacks persistent object references, has a very primitive naming service, no support for dynamic invocations (analogous to CORBA DII), and lacks a rich set of object services.

To conclude, Java RMI is easy to use for Java developers. Due to the shortcomings mentioned above, it is questionable, however, whether it is a suitable infrastructure for large, more complex systems.

6 CORBA products

This section presents an overview of CORBA products offered and their status by the end of 1997.

6.1 ORBs and Object Services

CORBA 2.0 compatible ORBs with IDL compilers are the basic CORBA product, and are available from several vendors. The most popular CORBA products in 1996 according to an IDC market research [6] are: Iona's Orbix (30 % market share), TCSI's Object Services Package ORB (21 %), BEA's ObjectBroker (16 %) and Visigenic's Visibroker (11 %). Other important ORBs are: NEO and Java IDL from Sun Microsystems and IBM's Component Broker. The leading ORB implementations are robust, because they have undergone several product cycles. Important programming language mappings and most operating systems are supported.

By the end of 1997 important CORBA services have been implemented, including naming, event, concurrency, transaction, trader and security. Naming and event service implementations have matured, while the transaction, trader and security implementations are first version implementations. Note though that transaction and security service implementations are mostly based on existing offerings (from DCE environments, etc.), hence they are fairly mature. Implementations of other services are in progress.

CORBA has emerged as a distributed object infrastructure for the Internet. During 1997 CORBA became Internet-ready in the sense that Java applets can interact with CORBA servers over the Internet using IIOP including lightweight Internet security support. Products include IDL to Java compiler, ORB implementation in Java, and IIOP over the Secure Socket Layer protocol, the latter providing authentication and encryption. Note that CORBA applications on the server side can also be programmed in Java. Further integration of CORBA and Java is evolving rapidly, see section 3.2.

The first versions of CORBA products targeted for large-scale applications were available by the end of 1997. This means provision of a middle-tier (see section 3.1) application platform that integrates the features of CORBA with scalable transaction management and the ability to access existing applications, databases and non-CORBA middleware. The objective is to provide a CORBA-based platform for large systems and mission-critical systems. System development and system management tools are also part of the offerings.

6.2 Integration with other products

Integration of CORBA with database management systems (DBMSs) in order to bring persistence to CORBA objects is supported, see section 3.3. Database adapters support adaptation between CORBA objects and database objects.

Adapters for ObjectStore, Versant and DB2 are available, and more adapters will soon follow. Database adapter frameworks that can be used to create new database adapters to DBMSs are also available.

TP monitors already exist which support scalable transaction management based on other message formats and transactions services than CORBA. The current trend is to integrate popular and well-proven TP monitors (Encina, Tuxedo, etc.) with CORBA, and the first integrations were available by the end of 1997. See section 3.4 for more information about transactions and TP monitors.

Interworking with non-CORBA middleware is being developed. OMG compliant Microsoft COM-CORBA gateways are available which enable COM-based applications to access CORBA objects in a transparent way. Proprietary gateways to IBM CICS and message oriented middleware (MOM) are also available.

The first environments for building, deploying and managing CORBA applications were available by the end of 1997. Vendor specific graphical development tools for CORBA are available. Also, popular development tools such as PowerBuilder, Rational Rose and IBM Visual Age are integrated with CORBA. System management tools are delivered by CORBA vendors implementing the CORBA system management facility specification, and also supporting a Simple Network Management Protocol (SNMP) interface in order to manage CORBA applications from popular management systems such as HP OpenView, Tivoli and Unicenter.

7 Future directions

OMG has planned a new version of the CORBA specification. The new version, which is called CORBA 3.0 and is due first quarter of 1998, will include updates and additions already made to the CORBA 2.0 specification, and several new and interesting enhancements that are on their way.

One of the new features that are expected to be included in CORBA 3.0 is the CORBA Messaging Service, which will provide the ORB with Message Oriented Middleware (MOM) functions. MOM is a key technology for a class of client/server applications. It allows general-purpose messages to be exchanged in a system using message queues. When using MOM, a client is allowed to make asynchronous, non-blocking requests. This means that the client does not have to be multithreaded, and clients will consequently be simpler and easier to maintain. MOM will allow clients and servers to run at different times, support nomadic clients and allow servers to determine when to retrieve messages off their queues. MOM will allow a looser coupling between applications than currently supported by ORBs which are often preferred for inter-application communication. On the downside, MOM does not support distributed transactions, because a transaction is broken into separate units of work.

OMG is currently extending the CORBA object model to allow objects with multiple interfaces, and to allow objects to be passed by value in operations. Another important ongoing initiative is to develop a component model for CORBA based on the Java component model, Java Beans. If the work is successful, it will be possible to view CORBA objects as Beans. This will in turn make it possible to manipulate general CORBA objects in graphical development tools (‘beanboxes’ in Java parlance), thereby reducing the complexity of creating CORBA applications.

Another enhancement that is expected to be included in CORBA 3.0 is server-side portable frameworks that will make CORBA servers portable from one ORB to another.

Within the CORBAfacilities component of the Object Management Architecture, much attention is given to mobile agents. A mobile agent is a CORBA object that can move its code and state across an IIOP network.

Within the domain areas, OMG members are currently specifying domain frameworks for several domains, including Electronic Commerce and Telecom. There is also work going on to build a generic business framework, called the Business Object Facility (BOF). The idea behind the BOF is to make it much easier to develop CORBA applications.

Readers who are interested in a more detailed account of the future development of CORBA, are referred to [3], on which the above outline is based.

8 References

1 Coulouris, G, Dollimore, J, Kindberg, T. *Distributed Systems : Concepts and Design.* Wokingham, Addison-Wesley, 1994. ISBN 0-201-62433-8.

2 Baker, S. *CORBA Distributed Objects.* Harlow, Addison-Wesley, Longman, 1997. ISBN 0-201-92475-7.

3 Orfali, R, Harkey, D, Edwards, J. *Instant CORBA.* New York, Wiley, 1997. ISBN 0-471-18333-4.

4 ACTS project AC048 (ReTINA). *BVPN Service Demonstrator Specification.* S.L., 1997. (AC048/TLN/WP5/DS/L/011 (D5.03).)

5 Mowbray, T J, Zahavi, R. *The Essential CORBA : Systems Integration Using Distributed Objects.* New York, Wiley, 1995. ISBN 0-471-10611-9.

6 Garone, S. *Middleware : 1997 Worldwide Markets and Trends.* International Data Corporation (IDC), 1997. (IDC report.)

9 Bibliography

Baker, S, Cahill, V, Nixon, P. Bridging Boundaries: CORBA in perspective. *IEEE Internet Computing,* 1 (5), 1997. http://computer.org/internet/.

ITU-T Recommendation M.3010. *Principles for a Telecommunications Management Network.* Geneva, 1996.

ITU-T Recommendation Q.1201. *Principles of Intelligent Network Architecture.* Geneva, 1993.

OMG. *Control and Management of Audio/Video Streams.* OMG RFP Submission. Revised submission. (OMG Document: telecom/97-05-07.)

OMG. *Notification Service Request For Proposal,* final version, 11.12.96. 1996. (OMG Document: telecom/97-01-03.)

OMG. *The Common Object Request Broker : Architecture and Specification,* revision 2.0. 1995.

Håkon Solbakken is Research Scientist at Telenor R&D, Kjeller, where he has been employed since 1986. He has been working with software specification and development in the field of network management and services. His interests include software development methods and tools, distributed object technology, and user interface design and implementation.

e-mail: hakon.solbakken@fou.telenor.no

Ole Jørgen Anfindsen is Research Scientist at Telenor R&D and an associate professor in the Software Engineering and Database Resarch Group, University of Oslo. He has been involved with database technology since 1982, and his research interests include databases, transaction models, object orientation, and CORBA. He is the Technical Representative for Telenor to the Object Database Management Group.

e-mail: ole.anfindsen@fou.telenor.no

Eirik Dahle is Research Scientist at Telenor R&D, Kjeller, where he has been working since 1988. He has worked with software specification and development in the field of network management and services. His interests include middleware, distributed object technology and database management systems.

e-mail: eirik.dahle@fou.telenor.no

Tom Handegård is Research Scientist at Telenor R&D, Kjeller, where he has been employed since 1990. He has been working with Intelligent Networks, B-ISDN signalling, and the TINA architecture. Currently, his main interests are within various aspects of distributed object-oriented systems, including CORBA and Java technology.

e-mail: tom.handegard@fou.telenor.no

Kjell Sæten is Research Scientist at Telenor R&D. After completing his masters thesis at Stanford University he spent several years developing software for seismic systems at Geco-Pracla. He joined Telenor R&D in 1997. His current interests are within distributed systems based on Java and CORBA technology.

e-mail: kjell.saten@fou.telenor.no

OMG. *Topology Service Request For Proposal*, draft 2, 16.01.97. (OMG Document: telecom/97-01-02.)

Schmidt, D. C. Distributed Object Computing. *IEEE Communications Magazine*, 35 (2), 1997, 42-44.

Vinoski, S. Distributed Object Computing With CORBA. *C++ Report*, 5 (6), 1993, 32-38.

Vinoski, S. CORBA : Integrating Diverse Applications within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, 35 (2), 1997, 46-55.

ZhongHua, Y, Duddy, K. *CORBA : A platform for Distributed Object Computing.* http://www.infosys.tuwien.ac.at/Research/Corba/archive/intro/OSR.ps.gz.

CORBA and Intelligent Networks (IN)

HELGE ARMAND BERG

This article discusses design of a CORBA/IN Interworking function (IWF). A static IN/CORBA IWF is proposed, residing on the INAP/TCAP layer, between Service Switch Function (SSF) and the Service Control Function (SCF). It is argued that the IWF should facilitate a smooth, real time interaction between the SS7 domain and the CORBA domain. In particular, the IWF should address harmonisation of the asynchronous, message oriented communication paradigm of the SS7 domain and the blocking, connection oriented paradigm of the CORBA domain. The IWF should also deal with the vital, low-level, real time, concurrent, event-based, queue-oriented aspects of the interaction and provide the IN service developer with high-level CORBA interfaces. The CORBA Event Service or the coming Message Service, enhanced with real-time and reliability requirements necessary for IN, is suggested for the interaction between the CORBA domain and the IN/CORBA IWF.

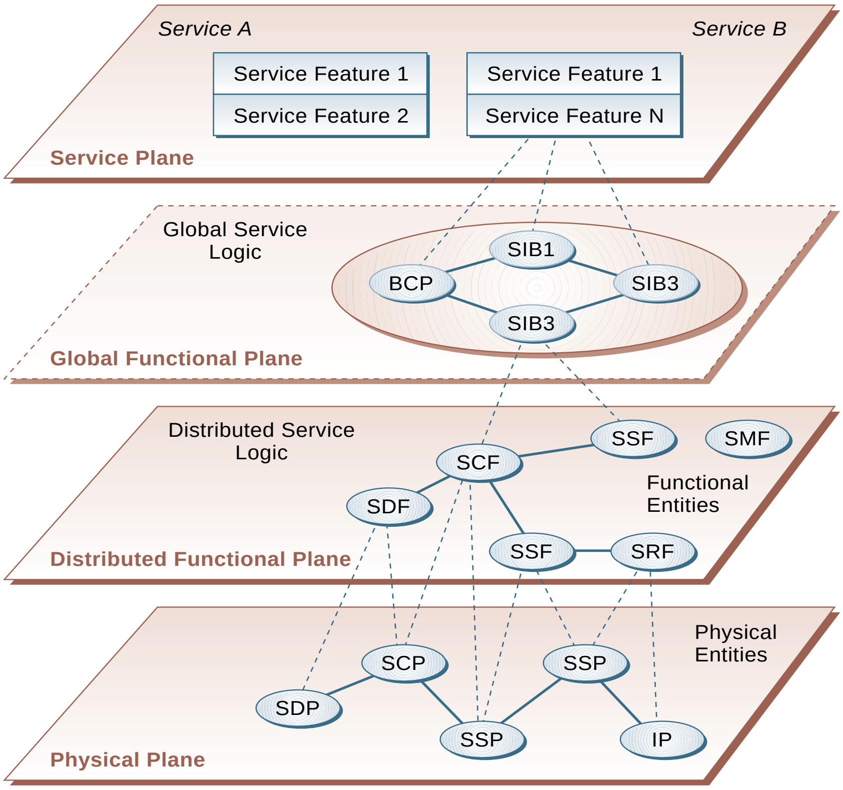


Figure 1 Use of the IN Conceptual Model to depict CS-1

Introduction

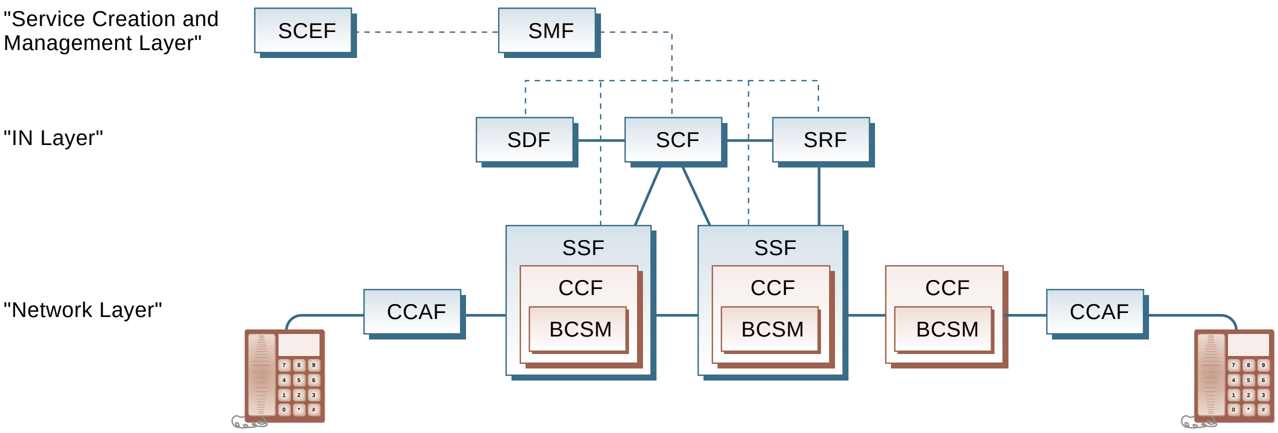
Rapid introduction of new services and a high degree of customisation are a necessity in the new, competitive environment where the telecom operators are confronted with a fast changing environment, deregulation and liberalisation. As the data and the telecom world converge, there is an increasing orientation in the telecom society towards an open software creation- and standardised computing-environment. There seems to be an increasing belief that middleware technology like CORBA can successfully be applied as an infrastructure in value-added telecom networks to meet the new challenges. CORBA technology has already been used in a number of non-real time application areas in the telecom world. This article addresses issues related to introduction of CORBA technology in the real time Intelligent Network domain. The design and implementation of an IN/CORBA Interworking Function (IWF) is outlined. An IN/CORBA IWF can be seen as a first migratory step towards TINA like networks and as an interworking unit between TINA like networks and traditional IN aware telecom networks. An IN/CORBA IWF can also be seen as an opening of the telecom vendor specific environment to achieve rapid creation, customisation and deployment of new services. Contrary to the telecom vendor controlled regime, an IN/CORBA IWF will facilitate third party delivery and competition of service creation environments and service components, both on software and hardware level.

The plan for this article is first to provide a short introduction of ITU standardisation of IN, followed by a discussion of what parts of IN are best suited for early introduction of CORBA. Design choices and design of an IN/CORBA IWF are outlined, and development, deployment and execution of services are presented in the context of an IN/CORBA IWF. The project is a co-operation between Telenor and the Irish company IONA Technologies, which so far is the most successful CORBA technology manufacturer. It is likely that other partners will be added later. The short term goal is to investigate the issues related to design and implementation of an industrial strength CORBA/IN IWF to produce an answer to a future OMG Request For Proposal on IN and CORBA interworking. The author is enjoying a stay with IONA Technologies.

Overview of Intelligent Networks

The basic idea behind the ITU standardisation of IN is to ease the introduction of new services by centralising the service logic in a few dedicated service nodes. The aim is to enable services to be added without costly upgrading of the switching hardware and software infrastructure. The service switching functions are separated from the service control software. The switch interrupts its processing at certain pre-determined points to query a remote server for service specific instructions. The switch can get several intermediate instructions before the final one, containing the routing address. Thus the service logic is removed from the switch and is relocated to a separate the service server. The service logic program can be viewed as a distributed application which partly runs on the switches and partly runs on the service server. Hence the switch does not have to be pre-loaded with service specific information.

Service independence is another key notion in IN, which addresses reusability



Service Excution
SSF Service Switching Function
SCF Service Control Function
SDF Service Data Function
SRF Service Resource Function

Call Handing
CCAF Call Control Agent Function
CCF Call Control Function
BCSM Basic Call State Model

Management and Service Deployment
SCEF Service Creation Environment Function
SMF Service Management Function
—— Call set up and invocation of IN
- - - - - Management and deployment of services

Figure 2 The "Service Creation and Management layer", the "IN layer" and the "Network layer" depicted in the IN CS-1 Distributed Function Plane. Services are created with the SCEF, deployed and managed with the SMF and executed in the "IN layer" controlling the "Network layer"

of components for building services. The IN Conceptual Model, INCM, is a modelling tool for describing the capabilities and characteristics of the architecture. ITU Standardisation of IN is seen as a stepwise evolution in what is called a Capability Set (CS). Typical CS-1 services are: number translation services, alternate billing services and screening services. Typical CS-2 enhancements are likely to be; end user interaction with service control outside the context of a call, mid-call interaction, mobility, IN interworking, IN management based on ITU-TMN management standards.

INCM consists of four planes which address different aspects of IN. The two upper planes focus upon the creation and perception services. The two lower planes address the IN functional and physical architectures. The end-user view of a service is defined in the Service Plane. Each service is compounded by one or more service feature(s) which are generic, reusable components. The IN service developer perspective is found in the Global Function Plane (GFP). The IN service developer builds an end-user service by combining Service Independent Building-blocks (SIBs) in the IN Service Creation Environment Function (c.f. Figure 2). Distribution issues are completely abstracted away from the Global Function Plane. The GFP views the telecom network as a virtual computer with service programs having virtual access to all computer programs and ignoring others. The Distributed Function Plane defines a distribution view of the IN in terms of units of network functionality called functional entities. The Service Control function (SCF) hosts the service logic programs. The Service Data Function (SDF) hosts service related data. The Service Switching Function (SSF) enables the triggering of services from the switches. The Service Management Function (SMF) supports the IN service management. The Specialised Resource Function (SRF) provides control and access to resources used in service provision in the intelligent network. The Intelligent Network Application Protocol (INAP) defines the interface protocol for IN control messages between the IN entities. The Physical Plane (PP) defines the real deployment of an IN. There are optional mappings from the Distributed Function Plane to the Physical Plane, but it is not possible to split one single functional entity among different physical entities.

An IN can be considered as a layer over any transport network (cf. Figure 2). The Call Control Agent Function (CCAF) is the initial access point to a network from the user terminal function. It passes call set-up and service request messages to the Call Control Function (CCF) for the basic call processing. To perform the basic call processing, the CCF maintains a Basic Call State Model (BCSM) both for calling and the called party for each incoming call. The BCSM is a finite state machine representing a set of different states of a call, like headset lifted off, number dialled, etc. The BSCM identifies logical points in the basic call processing at which IN service logic, located in the Service Control Function (SCF), is permitted to interact with the BCSM. The SSF provides a set of functions required for interaction between the CCF and the SCF. The IN physical entities communicate via the Intelligent Network Application Protocol (INAP), which relies on the Signalling System No. 7.

What functional entities to CORBAise?

There seems to be consensus in the IN/CORBA research society [2, 4, 5, 6, 7] that the SSF/CCF will be the last entities to be CORBAised (or being replaced by TINA aware switches), because the bulk of the organisational and technical investment resides here. Entities comprising the IN layer and the Service Creation and Management layer (cf. Figure 2) are considered to be better candidates for early CORBAtion. Complete or partwise CORBAtion of the IN-layer necessitates an IN/CORBA IWF (cf. Figure 3).

CORBA objects will replace the IN entities. IN control messages between the SSF and the IN layer are converted by the IWF to suitable CORBA object invocation(s). In the opposite direction CORBA replies, and method invocations are converted to IN control messages sent to the SSF/CCF. This article focuses on CORBAtion of the IN-layer, but an obvious step to make, given a CORBA-ised IN layer, is also to CORBAise the Service Creation Environment and Management of services deployed in a CORBA environment.

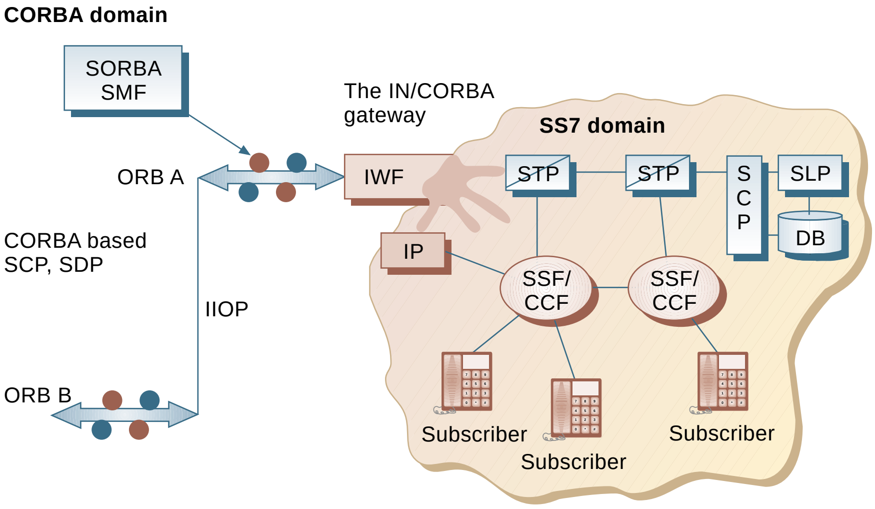


Figure 3 The IN/CORBA IWF and CORBAtion of the IN-layer

The IN/CORBA IWF; design choices

In this subsection different design choices of an IN/CORBA IWF are discussed.

- *IWF protocol layer:* In what protocol layer should an IWF reside? See next subsection.
- *Static or dynamic approach:* A static approach means using the CORBA Static Invocation Interface and static client stubs for invocation of CORBA objects. Such an approach implies that operations supported by the gateway must be known at compile time, though some configuration flexibility can be implemented. The dynamic approach refers to use of the CORBA Dynamic Skeleton Interface and the Dynamic Invocation Interface instead of pre-compiled client stubs. This approach gives more flexibility concerning what has to be known at compile time. The object invocations and protocol translation are constructed run-time.
- *Functionality located in the IWF:* Should the IWF just have pure protocol translation functionality or would it be beneficial to locate more functionality in the IWF?

INAP protocol layers

There are alternatives regarding on what protocol layer an IN/CORBA IWF should reside (cf. Figure 4).

INAP (Intelligent Network Application Protocol) uses the transaction capabilities of TCAP (Transaction Capability Application Protocol). TCAP is in turn based on ROS (Remote Operation Services) [10]. EURESCOM [2] proposes two variants of a simple, static IWF residing on the INAP layer. These approaches are claimed to be easy to implement. However, later work by EURESCOM proposes the more general and complex ROS gateway [4]. The AT&T reply [5,7] to the OMG Request For Information “Intelligent Networking using CORBA” [14] suggests a dynamic gateway at the ROS layer due to the widespread use of ROS in the telecom world. Also, a gateway in the SCCP layer has been proposed, because the layer above the SSCP is claimed to be vendor specific. This approach is not investigated in our project.

The discussion on the choice of protocol layer and static or dynamic gateway will be done in parallel because, in our opinion there is a close relation between the alternatives. For a gateway on the ROS layer a dynamic approach is an obvious choice, because this gateway will have application areas beyond the IN case. A static ROS gateway has less “value” than a static INAP/TCAP gateway, because the INAP and TCAP layer have to be rebuild. The value of a dynamic approach for an INAP/TCAP gateway is less obvious, because there are many predetermined aspects due the close relationship between the SSF and the SCF. Therefore, we concentrate our discussion on either a dynamic gateway on the ROS layer or a static gateway / “Interworking Function” on the INAP/TCAP layer. The use of the term Interworking Function instead of the term gateway will be motivated below.

Protocol layer and static or dynamic approach?

The Remote Operations Service Element (ROSE) protocol is an implementation of the Remote Operation Service (ROS) [10]. The ROS model provides a number of constructs to define the interaction between distributed objects using the information object class construct of the Abstract Syntax Notation One (ASN.1). The concepts of ROS and CORBA are mostly complementary. ROS is a very general communication paradigm based

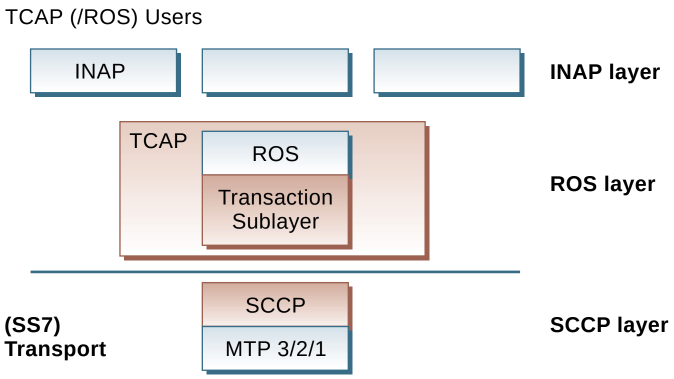
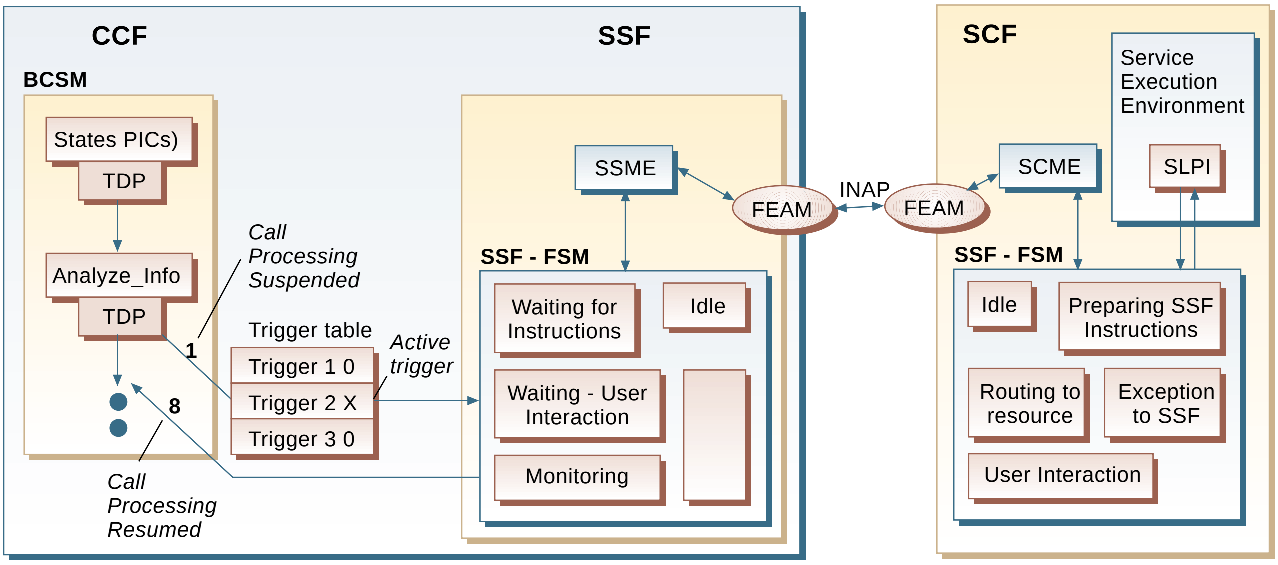


Figure 4 INAP protocol stack



CCF	Connection Control Function	SCF-FSM	Service Control Function - Finite State Machine
BCSM	Basic Call State Model	FEAM	Functional Entity Access Manager
TDP	Trigger Detection Point	SSME	SSF Management Entity
SSF	Service Switching Function	SCMF	SCF Management Entity
SCF	Service Control Function	SLPI	Service Logic Program Instance
SSF-FSM	Service Switch Function - Finite State Machine		

Figure 5 Relationship between the CCF/SSF and the SCF

on a simple request/reply model. Unlike a traditional client-server model, which defines the operation which the client may invoke on the server, the ROS model simultaneously defines both the client and the server aspects of a ROS object. In CORBA IDL the focus is on the server aspects of an object. [7] provides a "specification translation" between ROS and CORBA. To ease this translation, use of the TINA ODL based constructs 'required and supported interfaces' is suggested. The TINA ODL constructs 'required interface' correspond to the CORBA IDL interface for an object. The TINA ODL construct 'required interface' means an interface that an object is dependent on as a client. This construct is unavailable in CORBA at the moment. The INAP operations defined in Capability Set 1 [12] utilise the expressive power of ROS to a limited extent. A survey of the CS-1 specified INAP operations revealed that only one of 55 operations uses the "required interface" of ROS. This indicates an "overhead" of implementing the generic ROS gateway for the IN case. Due to the above reasons it will be experimented with a static INAP/TCAP Interworking Function.

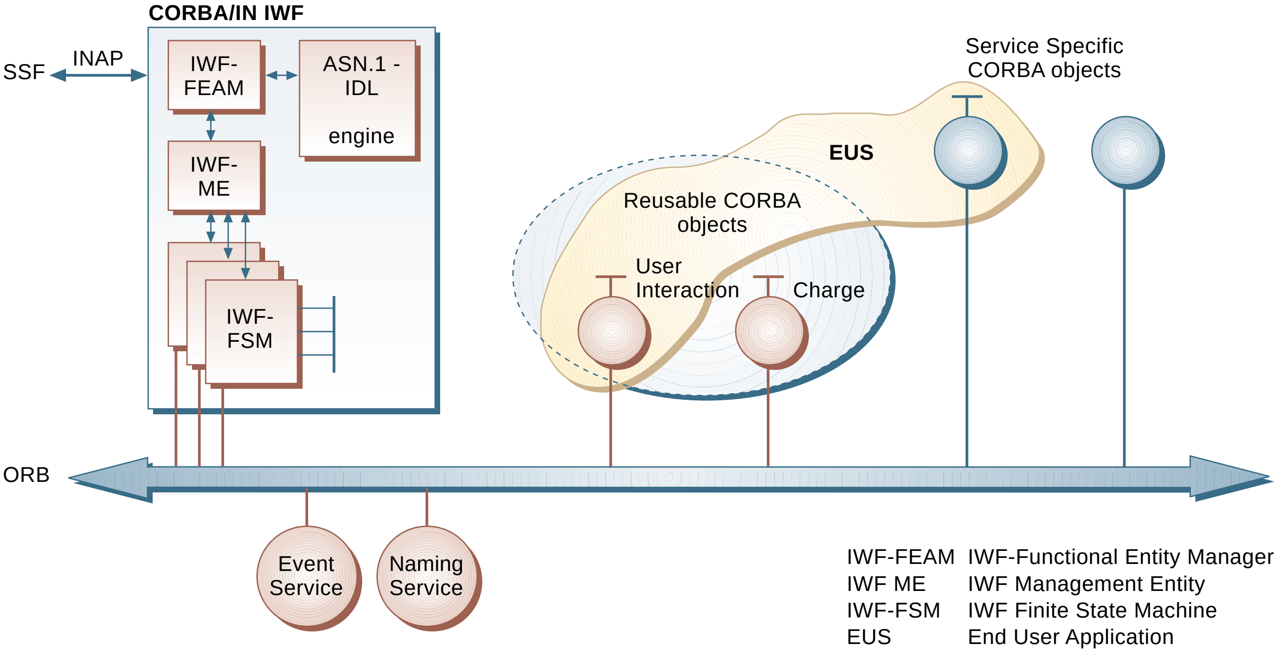
A thick or thin IWF?

What functionality should reside in the IWF is a major design choice. In order to get a better basis for this design choice, the interaction between the SSF and SCF, as defined in [11,12], is examined.

When a call/attempt is initiated by an end user and processed at an exchange, an instance of a BCSM is created (cf. Figure 5). As the BCSM proceeds, it encounters the detection points (DPs). If a DP is armed as a trigger DP (TDP), an instance of an SSF-Finite State Machine (SSF-FSM) is created. The SSF-FSM interacts with the SCF in the course of providing IN service features to users. It provides the SCF with an observable view of SSF/CCF call/connection processing activities, and provides the SCF with access to SSF/CCF capabilities and resources. It also detects IN call/connection processing events that should be reported to active IN Service Logic Program Instances (SLPIs), and manages SSF resources required to support IN service logic instances.

An example of an event that changes the state in the SSF-FSM is the firing of an armed TDP of type Request. This will change the state of the SSF-FSM from "Idle" to "Waiting for Instructions" and e.g. the INAP operation InitialDP will be sent to the SCF. The SSF-FSM in its present situation is awaiting further instructions from the SCF. At this stage a control relationship is established between the SSF and the SCF. A timer is started in the SSF in order to avoid excessive call suspension time for this specific call. The SSME maintains the dialogues with the SCF on behalf of all instances of the SSF-FSM. The Functional Entity Manager (FEAM) provides the low level interfaces for establishing and maintaining the interfaces to the SCF, and passing and queuing messages to the SSME and the SCME. When the InitialDP INAP operation is received by the SCF, an instance of an SCF Call State Model (SCF-FSM) is created, and the relevant SLP is invoked. Multiple requests (INAP operations) may be executed concurrently and asynchronously by the SCF. This motivates the need for a single entity that performs the tasks of creation, invocation and maintenance of the SCF-FSM objects. This entity is called the SCF Management Entity (SCME). In addition to the above tasks, the SCME maintains the dialogues with the SSF on behalf of all instances of the SCF-FSMs through the FEAM.

We summarise the characteristics for the interaction between SSF and SCF: event based, non-blocking, concurrent and queue-oriented exchanges of INAP operations, and an infrastructure consisting of managers and Finite State Machines that prevent the SLP dealing with such issues.



IWF-FEAM	IWF-Functional Entity Manager
IWF ME	IWF Management Entity
IWF-FSM	IWF Finite State Machine
EUS	End User Application
ORB	Object Request Broker

Figure 6 The IN/CORBA Interworking Architecture

Significant traits of CORBA object invocations are synchronous or deferred synchronous, blocking and connection-oriented. The paradigm differences between the CORBA and the SS7/IN should preferably be harmonised by the IN/CORBA IFW. Useful tools for achieving this seems to be CORBA Services, like the Event Service or the coming Messaging Service. These services facilitate asynchronous, message based CORBA object interactions in a de-coupled manner, like the interacting entities in the SS7 domain. Implementations of the above mentioned CORBA Services with the required reliability and real-time characteristics for the IN case can serve as a means for harmonisation between the domains.

The Event Service is intended as an optional extension to an ORB to allow clients to communicate with application objects using events. Suppliers generate events, and consumers receive them. The Event Service defines a de-coupled communication style. It defines an indirect communication style using event channels, where both push and pull event delivery are supported [15]. The coming Messaging Service is likely to support asynchronous invocation, contain quality of service features such as lifetime of a request, reliability, server side queue management, load balancing and routing. EURESCOM promotes in [4] an extension of the Event Service to address real time and the fault tolerant requirements of IN. Use of such an Event Service is claimed to facilitate a flexible, extensible, “component plug-in” oriented architecture for building services.

As has been motivated above, a thin IWF which only covers syntactic protocol translation between the SS7 domain and the CORBA domain may push too much responsibility on the CORBA server objects in order to deal with queue management, concurrency, real time, and the event based interaction with the SSF. Therefore, a thicker IWF is investigated in the project, and the term Interworking Function is preferred to gateway, because in our approach we want to put more than protocol translation functionality to the IWF.

The IN/CORBA IWF

As explained in the previous sections the design of a thick Interworking Function on the TCAP/INAP layer is proposed. In the following text the term End User Service (EUS) refers to an End User Service implemented in the CORBA domain. The EUS corresponds to the SLPI in the IN domain.

The following components to the SCF’s FEAM and SCME are necessary in the CORBA/IN IWF:

- IWF-Functional Entity Access Manager (IWF-FEAM)
  - Provide access to other non-CORBA Functional Entities (SRF, SCF, SDF)
  - Translate incoming/outgoing INAP operations to/from CORBA types using the IDL-ASN.1 engine
  - Provide queue-management of incoming and out-going INAP operations
- IWF-Management Entity (IWF-ME)
  - Create, maintain and delete the IWF-FSM for each call that needs processing of the IN-layer
  - Maintain the dialogues with the SSF on behalf of all SCF-FSMs trough the IWF-FEAM
- The IWF-Finite State Machine
  - Interact both with the SS7 domain through the IWF-ME and the CORBA domain through an Event Service
  - Resolve the CORBA reference to the EUS using the Naming Service
  - Start an EUS instance asynchronously with an Event Service
  - Keep the necessary CORBA references
  - Provide a set of IDL operations which correspond to INAP opera-

tions of interest to the EUS to invoke (see example in the text below)

- "Listen to" invocation from the EUS and the SSF and take necessary actions
- Maintain timers to supervise the EUS and the interaction with the SSF.

The translated INAP operation will be directed to the correct instance of the IWF-FSM by the IWF-ME based on the CALL-ID if there exists an IWF-FSM for this call. Otherwise, a new IWF-FSM instance is created. The IWF-FSM will resolve CORBA object references based on the parameters of the translated INAP operation using the Naming Service. The IWF-FSM will contain the fine-granular timer semantics that are characteristic for SSF/SCF interactions. The EUS will be invoked asynchronously by an Event based Service, initiated from the IWF-FSM. The IWF-FSM will be able to listen to asynchronous events or requests from the EUS and the SSF and take actions based on them. The IWF-FSM will contain IDL interfaces which represent the INAP operations which are of interest to an EUS to invoke. A typical example is the INAP operation PromptAndcollectUserInformation, which is used to collect more information from an end user, such as a pin code. If this operation is invoked on the IDL interface of the IWF-FSM, then the state is changed to "User Interaction" and the operation is sent to the SRF (cf. Figure 2). The EUS does not need to stop its executions as this happens. A response from the SRF will be reported back to the EUS and through the IWF-FSM asynchronously. Error situation in the EUS will be reported back to the relevant IWF-FSM through the Event based Service, as well. Errors and time outs in the SSF are received by the relevant IWF and necessary actions, like clearing the resources in the CORBA domain, will be performed.

Detailed requirements to the IN/CORBA IWF can be found in [2, 4] and elsewhere. Some additional design requirements are summarised below:

- The IWF should be independent of the implementation of EUS. There should be no need for recompiling the IWF when new services are introduced.
- An open, precise interface is required between the IWF and the CORBA domain in order to facilitate third party delivery and competition of Service Creation Tools. Re-implementation of CORBA objects with SIB like functionality can be considered as suggested in Figure 6. Another possibility is to implement TINA inspired objects interacting with IN. This has been examined in [6].
- The IWF should be modular with regard to different INAP protocol stacks. The modification on the IWF in case of adjusting the IWF to different vendor protocol INAP stacks should be minimal, and if possible, such adjustments should be configurable without recompiling.

References

1 Magedanz, T, Popescu-Zeletin, R. *Intelligent Networks : Basic Technology, Standards and Evolution.* London, Thomson Computer Press, 1996. ISBN 1-85032-293-7.

2 EURESCOM. *CORBA as an Enabling Factor for Migration from IN to TINA.* (A EURESCOM-P508 Perspective (Final), Annex 5.)

3 ALCATEL. *Intelligent Networking using CORBA.* 1997. (ALCATEL's Response to the OMG Request for Information. April 1997, Reference ULT/C/97/0121.)

4 EURESCOM. *Introduction of Distributed Computing Middleware in Intelligent Networks.* (A EURESCOM-P508 perspective, OMG Doc. Number orbos/97-09-11.)

5 AT&T. *Design of a ROS : CORBA Gateway for Inter-operable Intelligent Networking Applications.* 1997. (AT&T Response to the OMG Request for Information, April 1997, OMG – telecom/97-04-01.)

6 Herzog, U, Magedanz, T. From IN toward TINA : Potential Migration Steps. In: *International conference on intelligence in services and networks : technology for cooperative competition, IS&N, Cernobbio.* Berlin, Springer, 1997.

7 Mazumdar, S, Mitra, N. ROS-to-CORBA Mappings: First Step towards Intelligent Networking using CORBA. In: *International conference on intelligence in services and networks : technology for cooperative competition, IS&N, Cernobbio.* Berlin, Springer, 1997.

10 ITU-T. *Information technology – Remote Operations: Concepts , model and notation.* ITU-T Rec. X.880, 1994. ISO/IEC 13712-1:1995.

11 ITU-T. *Distributed functional plane for intelligent network CS-1.* Geneva, ITU, 1995. (ITU-T Rec. Q.1214.)

12 ITU-T. *Interface Recommendation for intelligent network CS-1.* Geneva, ITU, 1995. (ITU-T Rec. Q.1218.)

13 ETSI. *Core INAP.* 1997. (ETSI Document ETS 300 374.)

14 OMG. RFI on *Issues concerning Intelligent Networking with CORBA.* (Nilo Mitra, AT&T, OMG - telecom/96-12-02.)

15 OMG. *CORBAservices : Overview.* (Formal/97-02-06: CORBA services specification.)

Helge Armand Berg has been working as Research Scientist at Telenor R&D since 1991 in the areas of TMN, TINA, CORBA and distributed database systems. Since June 1997 he has been working on IN and CORBA in a co-ordinated project between Telenor and IONA Technologies, Dublin. He has a particular professional interest in the CORBA technology.

e-mail:
hberg@iona.com

Jada: Java, Architecture, Development and Application

ARNE SOLEVÅG HATLEN, ØYSTEIN MYHRE,
BIRGER MØLLER-PEDERSEN AND STIG JARLE NESS

Introduction

The architecture presented here has been developed as part of the Jada project at Telenor R&D. The original objective of the project was to evaluate the use of Java/Internet technology for Telecommunication Management Network (TMN). The Jada architecture and tools for making this kind of application have uncovered additional results from the project.

This presentation of Jada has its focus on architecture. For details on the project and on the prototype GDMO/Java tools developed as part of the project, please consult [2].

The following dimensions of architecture will be covered:

- Distributed Object Architecture
- Application Architecture
- Architecture of Development Tools.

Jada Architecture Overview

A Jada application is organised as a set of interacting server and client systems. Each server system is organised as a naming tree of Java objects, while clients are Java applets. This is shown in Figure 1. Client-server and server-server interaction is provided by means of Java Remote Method Invocation (RMI), so server objects are characterised by remote interfaces. Systems are either associated with and used by a client user application in terms of an applet, or are used by each other; in both cases by means of Java remote method invocation between objects. Systems may either be independent systems or just components of a larger system.

The ideas behind the architecture are:

- Operator/user applications can be made readily available through widespread browser technology (Java Applets).
- The Java technology, including distribution by means of RMI, can be used for real application development, and not just for making fancy applets as part of web-pages.

These elements of the architecture are not unique for Jada – but is shared by most Java applications involving RMI and applets.

However, Jada applies the object model of GDMO [5] and maps this into Java object models. The GDMO approach means that each object is characterised by a set of packages, each having attributes and actions. Attributes are specified to have values according to ASN.1 types, and may be specified as GET or GET-REPLACE operations. Actions are specified with parameters and results in terms of ASN.1 types. Clients and objects in other servers may set and get attributes and request objects to perform actions. In addition, objects may emit notifications.

Objects are named within a system by means of name bindings. A system has as its root an object of class system. Each object named by another superior object has an attribute that is used for naming.

The Jada development environment contains, in addition to a pure Java development environment, a set of GDMO development tools that help in producing Java representations of GDMO specified object models.

As shown in Figure 2, the Java implementation is generated from GDMO spe-

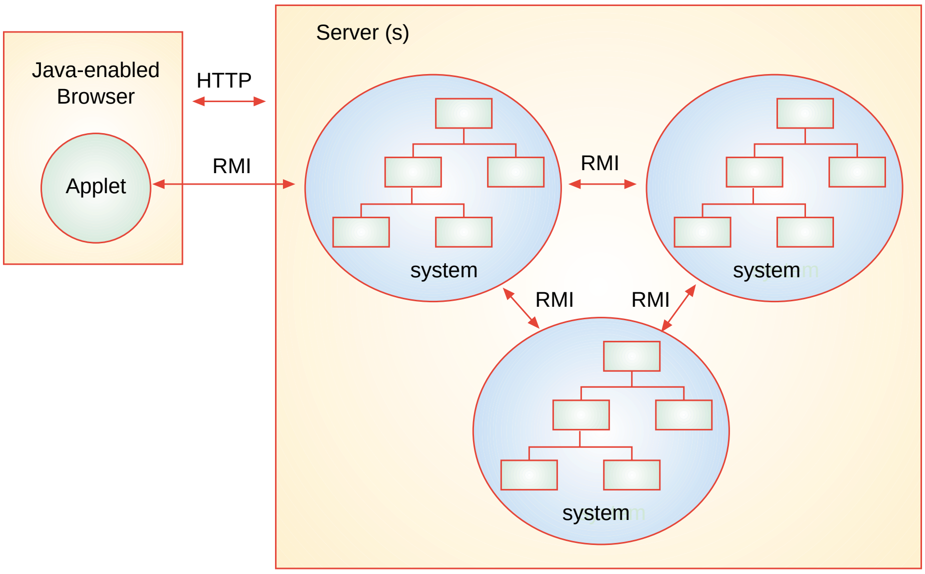


Figure 1 A Jada application consisting of a set of server systems and a client applet

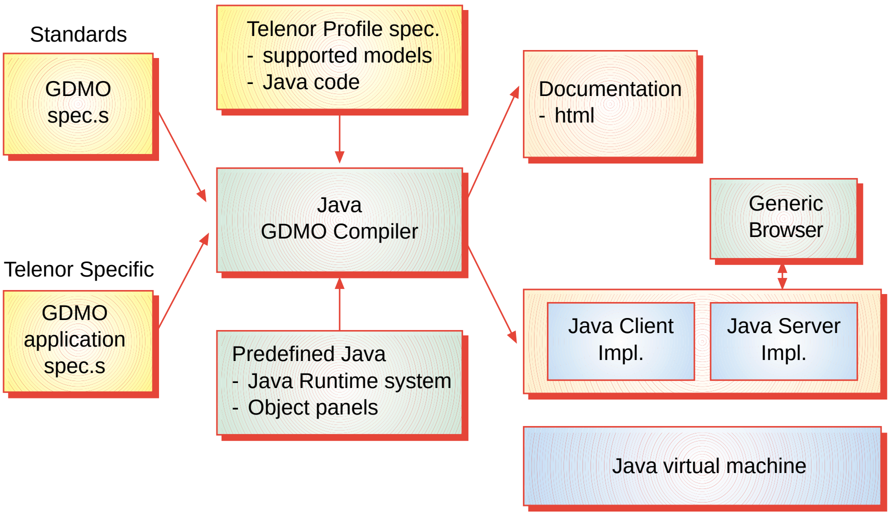
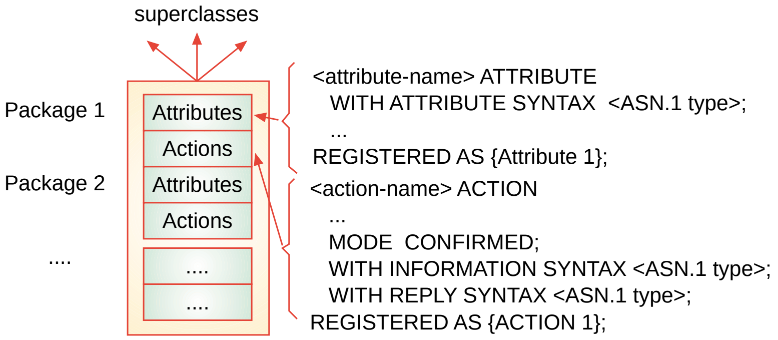


Figure 2 Architecture of Jada Development Tools



```
<attribute-name> ATTRIBUTE
  WITH ATTRIBUTE SYNTAX  <ASN.1 type>;
  ...
REGISTERED AS {Attribute 1};
<action-name> ACTION
  ...
  MODE  CONFIRMED;
  WITH INFORMATION SYNTAX <ASN.1 type>;
  WITH REPLY SYNTAX <ASN.1 type>;
REGISTERED AS {ACTION 1};
```

Figure 3 GDMO class objects with packages, superclasses, attributes and actions

cifications. These include GDMO specifications of standard object models in the domain; in addition comes the more application specific specifications. Usually, GDMO specifications are very generic, with optional packages covering a wide variety of cases. Profile specifications provide the subset that is used. In addition, the profile specifications may include Java code to complement the skeleton Java code generated from the GDMO specifications. The GDMO compiler will among other things generate the method name and parameters for an action specified in GDMO, while the complementary Java code in the profile specification will provide the implementation of this action. The generated Java code is running on top of a Jada Runtime System that in turn is run by the Java virtual machine. In order to present and enter values in attributes of objects, a Panel is defined for each class. In addition, a generic Browser provides means for navigating in the naming trees of the systems.

Distributed Object Architecture

The basic execution architecture of Jada is an architecture of distributed Java RMI objects. For this presentation it is not important which underlying protocol that is used. It could be: HTTP, a special RMI protocol, a protocol for a combined CORBA/RMI approach, or another protocol. For a discussion of performance and security, the choice of protocol may however be important.

The distributed object architecture comes about by mapping the GDMO object model shown in Figure 3 into a corresponding API on top of Java/RMI:

- GDMO Object classes are mapped to Java interfaces and implementation classes.
- GDMO attributes are mapped to set- and get methods of the Java interfaces and implemented by variables and implemented set- and get methods in the implementation classes.
- GDMO actions are mapped to methods of the interfaces and implemented in the implementation classes.
- GDMO notifications are mapped to a special method that treats the notification and possibly generates event reports by means of remote method invocations on destination objects.
- GDMO packages are mapped to interface types with methods representing the contents of the packages.
- GDMO behaviour specifications are mapped to documentation only (in HTML).
- GDMO name bindings are mapped to create methods in the naming classes and for each class to a general Container interface specifying the containment relationship for objects of the class.
- ASN.1 types are mapped to data type classes in Java, and these are then used to specify variables in the implementation classes and parameters to methods.

One may argue that Managed Object Classes should become Java classes, but in fact, Managed Object Classes really just define how the manager views the information in the MIB, and this is exactly the purpose of interfaces in Java.

As is indicated in Figure 5, a Managed Object Class ‘C’ is mapped to an interface ‘C’ that extends interfaces corresponding to both superclasses and packages, and the corresponding implementation class ‘CImpl’ extends ManagedObjectImpl and implements the interface ‘C’.

The fact that the managed objects may be named in the containment tree is reflected by the fact that ManagedObjectImpl extends RemoteContainerImpl. This is not shown in Figure 5.

RMI

Remote Method Invocation (RMI) enables objects in one Java Virtual Machine (VM) to seamlessly invoke methods on objects in a remote VM. To invoke a method on a remote object the Java program needs a reference to the object. A remote reference is obtained either as an argument or return value, or by means of the simple name service provided by RMI. To lookup a remote object the name of the object and the location (hostname) must be known.

The architecture of RMI consists of three layers, as shown in Figure 4:

- Stub/Skeleton Layer: Client-side stubs and server-side skeletons
- Remote Reference Layer: Reference/invocation behaviour
- Transport Layer: Connection set up and management, and remote object tracking.

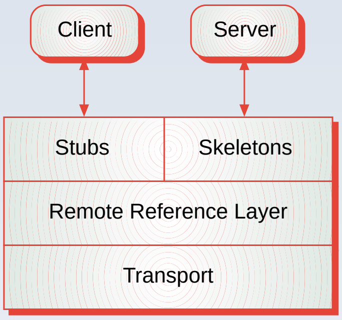


Figure 4 The layers of RMI

A remote object implements one or more remote interfaces, which are pure Java interfaces that declare the methods of the remote object. A method invocation on a remote object has the same syntax as a method invocation on a local object.

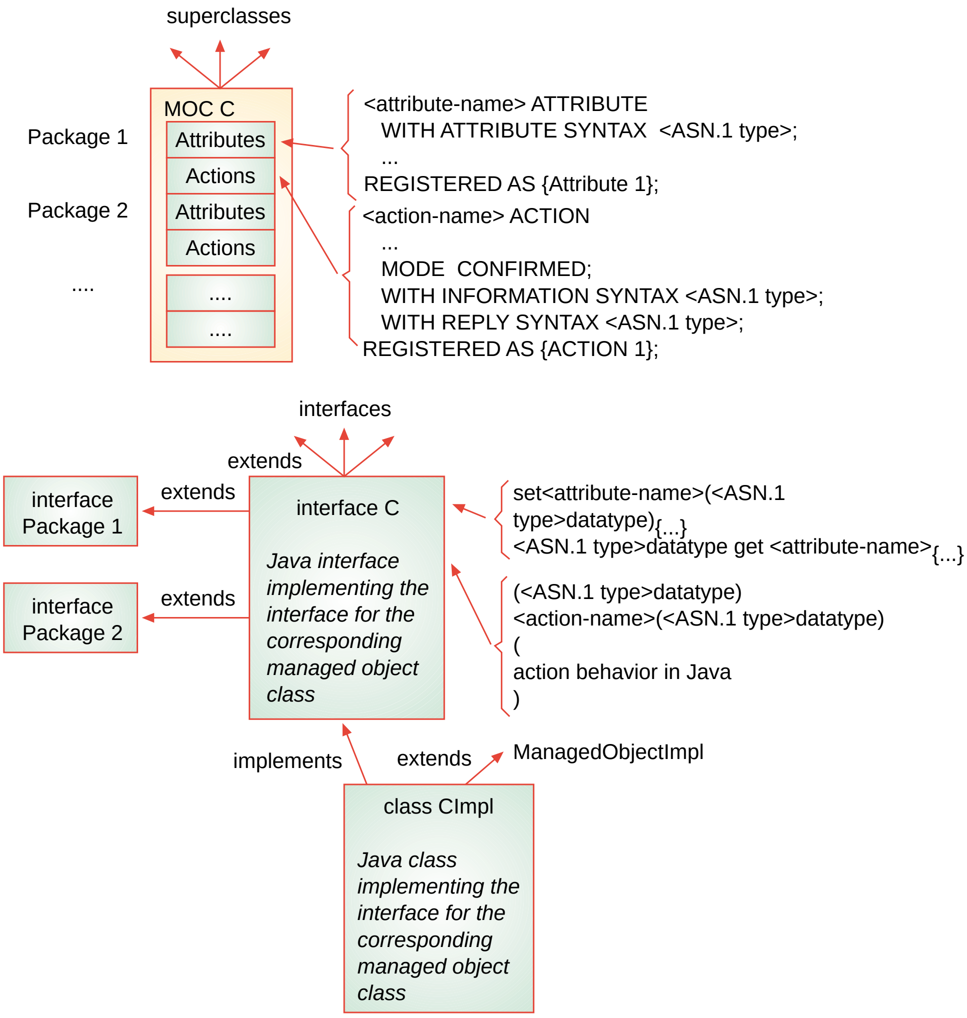

superclasses
MOC C
Package 1
Attributes
Actions
Package 2
Attributes
Actions
....
....
....
<attribute-name> ATTRIBUTE
WITH ATTRIBUTE SYNTAX <ASN.1 type>;
...
REGISTERED AS {Attribute 1};
<action-name> ACTION
...
MODE CONFIRMED;
WITH INFORMATION SYNTAX <ASN.1 type>;
WITH REPLY SYNTAX <ASN.1 type>;
REGISTERED AS {ACTION 1};
interfaces
extends
interface Package 1
extends
interface C
Java interface implementing the interface for the corresponding managed object class
set<attribute-name>(<ASN.1 type>datatype){...}
<ASN.1 type>datatype get <attribute-name>{...}
interface Package 2
extends
(<ASN.1 type>datatype)
<action-name>(<ASN.1 type>datatype)
(
action behavior in Java
)
implements
extends
ManagedObjectImpl
class CImpl
Java class implementing the interface for the corresponding managed object class

Figure 5 GDMO to Java mapping

Application Architecture

The architecture of objects within server- and client systems is normally not considered a part of a distributed object model, but is left to whatever possible architectures being supported by the actual implementation language. This is not the case with Jada. In addition to the basic structuring of systems by means of name bindings mentioned above, Jada applies the following classification of objects (or in some cases just aspects of objects) into the following categories (shown in Figure 6):

- *Model specific* objects/aspects of objects, which come from an object model of the domain
- *Application specific* objects/aspects of objects, which have to do with the application specific functionality of a given application and not of the domain
- *Interface specific* objects/aspects of objects, that define either interfaces to other systems or user interfaces.

The classification into model, application and interface does not have to classify whole objects, but also aspects of objects, i.e. one object may have both model and application specific aspects. In order for the classification to work for aspects, it is required that the specification of the different aspects of the same object class can be isolated and identified. Depending on the situation, we will talk about application objects or application specific aspects. Details of this classification can be found in [1].

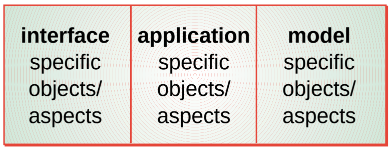

interface specific objects/ aspects
application specific objects/ aspects
model specific objects/ aspects

Figure 6 Object classification categories

There are three motivations for the model-application-interface classification:

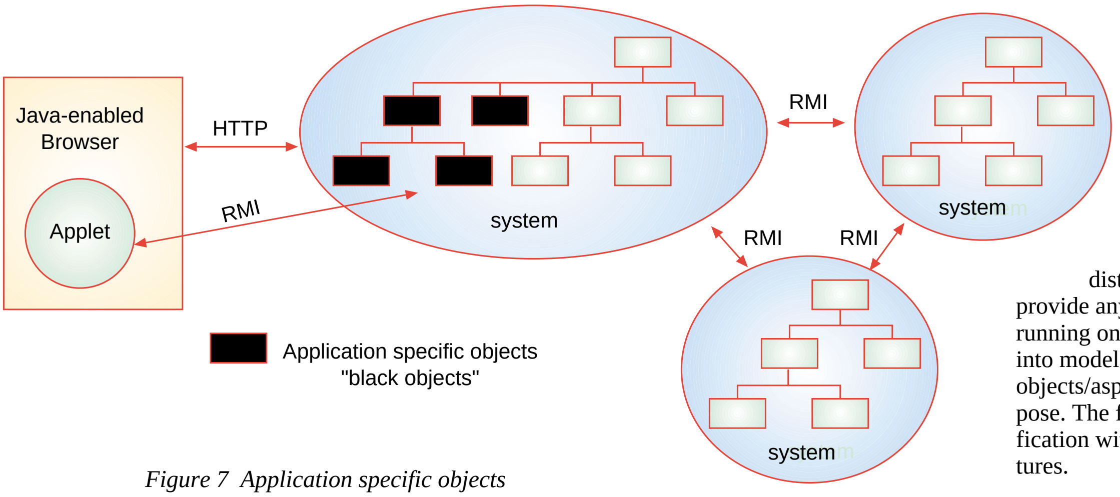


Figure 7 Application specific objects

- It helps in classifying requirements: An application will have requirements on
  - which application domain to cover
  - which functionality to provide
  - which users and other systems to interface to.
- It helps in stabilising a given design: The objects coming from a domain model is more stable than the objects providing the functional requirements, as these may change with changing requirements.
- It supports reuse: The reason for identifying the model objects is that the corresponding classes have a large potential for (re)use within other applications in the same domain.

An object-oriented analysis of a domain will result in a domain object model. When an application in the domain is implemented in terms of objects, it is obvious to start with objects that come directly from this analysis.

The domain of Jada (network and service management) has several domain object models covering network elements and networks. These domain object models are standardised and specified in GDMO. The model specific objects of Jada applications are produced by mapping these GDMO specified object models to corresponding Java object models.

Most object-oriented analysis methods recommend domain object modelling. In contrast to most of these, where domain object models consist of classes of ‘passive data objects’, the typical domain object models for Jada include managed objects that have actions.

Application specific objects are objects that are particular for the given application. While requirements on the application that have to do with the domain may be associated with the model specific objects, functional requirements may lead to application specific objects/ aspects.

Giving the model specific objects (and classes) of an application,

- requirements that naturally can be associated with the model specific objects lead to the introduction of application specific aspects of the given model specific objects, e.g. by defining subclasses with additional methods that represent application specific functionality
- requirements that cannot naturally be associated with the model specific objects lead to the introduction of separate application specific objects.

As the Jada development tool makes it possible to associate Java code with GDMO specified classes, it is possible to specify these application specific objects in GDMO and let them be contained as subordinate to the root system object, see Figure 7. They may either be made visible through naming attributes or just be ‘black objects’ that are there to provide the application specific functionality.

As mentioned above, most distributed architectures do not provide any structure for parts of systems running on servers, and the classification into model, application and interface objects/aspects is introduced for this purpose. The following compares this classification with two- and three-tier architectures.

Traditionally, client-server applications were mostly based on two-tier architectures; either with all the application logic on the desktop, the server running only the database, or with a slightly thinner client and some application logic on the server. The main advantage of the two-tier architecture is its simplicity; it is quite straightforward to decide which functions go where.

In three-tier architectures the first tier contains the clients. The clients typically provide a user interface and some local processing. The second tier contains the application logic (often called ‘business objects’), while the third tier contains the database(s). The clients never access the third tier directly. The middle tier makes abstractions of the resources in the third tier.

Business objects within the second tier may themselves have client-server relationships (multi-tier architecture).

The back-end tier contains databases and legacy applications. The middle tier objects communicate with the databases and external applications in the third tier.

The three-tier architecture has some resemblance to the model/application/ interface classification. Business objects may be introduced to capture application specific aspects. While the three-tier architecture is introduced for ‘physical’ reasons (load distribution, independence of data representation, etc.), the model/ application/interface classification is introduced in order to structure a large number of objects independent of whether they are part of clients, servers or databases. When a three-tier architecture is applied to an application that also classifies by means of model/application/interface, there may be interface specific objects in the middle tier and application specific objects in the first tier. The third tier will probably mostly have model specific objects (the data base objects of a domain object model), but the middle tier may also have model objects. As an example, a GDMO-specified Management Information Base (MIB) may be represented both as database objects in the third layer and as 'agent' objects together with the other application specific objects in the middle layer.

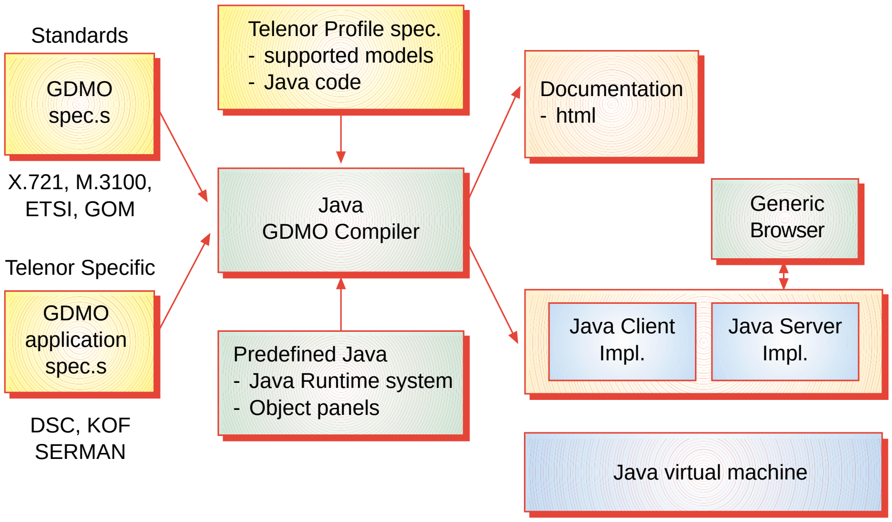


Figure 8 Jada Development applied to example systems

Architecture of Jada Development Tools

The special thing about the Jada development architecture is that it is centred around GDMO specifications and that as many as possible of the applications are generated from these. Parts of the Java implementation is generated from GDMO specifications:

- standard object models in the domain, and
- application specific specifications.

In Figure 8 the development architecture is illustrated with the GDMO specifications that have been used in the example system.

Profiles

Usually, GDMO specifications are generic, with optional packages covering a wide variety of cases. The profile specifications provide the subset that is used for the specific cases. The profile also specifies the system root which is normally not specified in existing standard object models. Profiles are specified in a notation similar to GDMO. No effort has been put into making this notation a complete language in the sense that behaviour of objects can be specified. See Figure 9.

Code

GDMO specified models include specification of behaviour in terms of informal text only. For the purpose of generating server systems according to a GDMO model, a simple mechanism for associating code with the GDMO specifications has been developed. In order not to change the GDMO specifications, the code is provided in separate specifications. The profile specifications are the basis for the implementation, and it associates code to each class, set/replace method and action of the profile, see Figure 10.

Jada Runtime System

The Jada Runtime System consists of a number of Java interfaces and classes that implement basic properties of managed objects in accordance with the GDMO standard X.722 [5].

- Attribute types in terms of ASN.1 are supported by a set of Java classes implementing the ASN.1 types. In addition, there are Panel object classes defining user interface components for all these types. Each panel shows the contents of the corresponding value object, and provides both an editable and a view-only version.
- Name bindings are supported by Java interface types and implementation classes that provide navigation in MIBs.
- The capability of sending event reports is supported by an EventReporter interface and of receiving event reports by the EventReportDestination interface are supported.

```
logProfileBehaviour BEHAVIOUR DEFINED AS
" This interface defines the Telenor
" concrete MIB interface to the X721.Log
" managed object class. ";

Log PROFILE
INCLUDES X721.log;
  SUPPORTS CONDITIONAL PACKAGES
      //FROM: X721.log:
        X721.finiteLogSizePackage,
        X721.logAlarmPackage,
        X721.availabilityStatusPackage,
        X721.duration,
        X721.dailyScheduling,
        X721.weeklyScheduling,
        X721.externalScheduler,
      //FROM: X721.top:
        X721.packagesPackage;
      //X721.allomorphicPackage;
      //NOT SUPPORTED
    SUPPORTS NAME BINDINGS
        X721.log-system;
  BEHAVIOUR logProfileBehaviour;
REGISTERED AS {telenorProfileID 999};
```

Figure 9 Example Profile Specification

The JADA prototype environment contains a generic browser for inspecting MIBs. It is generic in the sense that it is independent of the information model being inspected. The browser is made by use of the Java Reflection API. With this API it is possible to get information about any object, such as information

```
//$CODE=$PRF:Telenor.KOF.Kof#DECLARATION
     // CODER:Attached DECLARATION Code

/**
* This variable carries the reference to the KOFagent in Sermas.
**/
  Telenor.KOF.KofAgent kofAgent;


//$CODE=$PRF:Telenor.KOF.Kof#ACT:KOF.invoicing
         // CODER:Attached ACTION Code
         { KOF.createInvoiceParam cIP = new KOF.createInvoiceParam();
           // assign to cIPs attributes from argument
           BillingImpl aBillingImpl= (BillingImpl)billing;
           cIP.setAccountId(
             (aBillingImpl.createOrFindAccount(
               argument.getCustomerId())).getAccountId());
           cIP.setPeriod(argument.getPeriod());
           messageProcessing.createInvoice(cIP);
       }
```

Figure 10 Example Code Specification

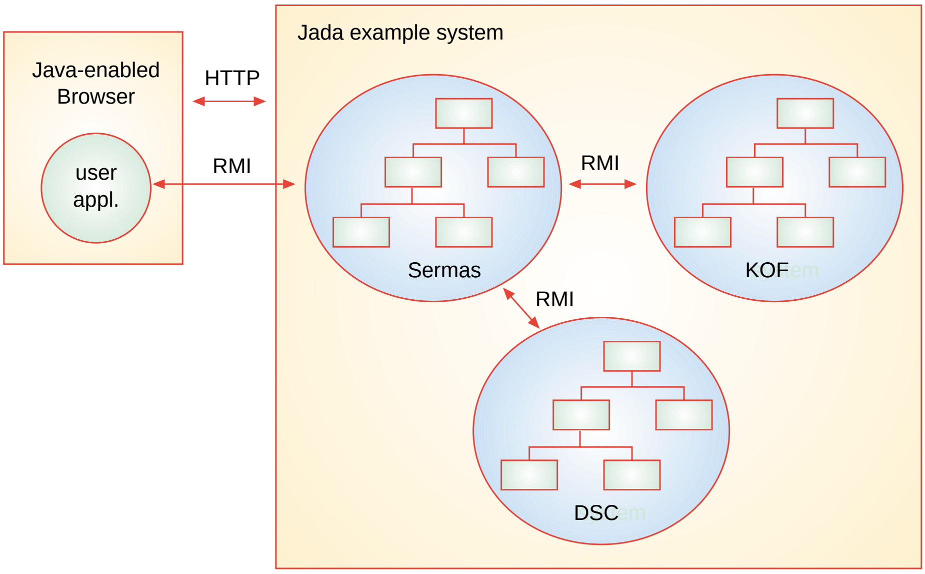


Figure 11 Jada Example System

about methods, constructors, fields, etc. Due to the mapping of GDMO to Java, the Browser needs a little help. For instance, it needs help to identify which methods that are operations on attributes and which are in fact actions defined in GDMO. In order to solve this problem, all objects can be asked for their GDMO-attributes, actions and name bindings.

The Browser will then use reflection to get the parameters right.

The use of the Reflection API together with the DataPanels enables the Browser to display all objects, regardless of type. In addition to viewing an object, it is also possible to change its attributes by invoking the appropriate set-methods.

Application of the Jada Architecture

In order to illustrate the architecture issues of Jada, a prototype Jada application is presented. The domain of this prototype application includes both GDMO specified network models and application models. Among the application objects are the customer class and order class. In addition, the example application implements an interface to the invoice system of Telenor (KOF).

The distributed object architecture

The example Jada application consists of three server systems (Sermas, KOF and DSC). The distributed object architecture of the example application is shown in Figure 11.

- Sermas is a Telenor-specific system supporting user configuration of SDH-lines and status reporting about a customer’s subscribed lines. Sermas relies on the DCS system to provide connections to Sermas. Sermas handles information about Customers and Customer Subscriptions, a Customer’s Access Points (CAPs), as well as information about SDH leased-line services. For details, see the Sermas specification in [3]. The GDMO specification for Sermas is generated from the corresponding OMT specification from the Sermas Requirements Specification.
- DCS is the network management system providing the SDH leased-lines to Sermas. The SDH network equipment and connections are registered, monitored and maintained in the DCS. The system is based on a GDMO specification, which in turn builds on ITU-T M.3100 and ETSI GOM.
- The KOF application is based on a small subset of KOF specified in GDMO for the purpose of the demonstration. To populate and interrogate the different MIBs, a generic browser is used. The Browser navigates through the data by using the naming hierarchy defined in the MIBs. The only client application with a user interface (and even that is kept at a minimum) is the Sermas operator application, while the other applications are interfaced by means of the browser.
- The user application supports an operator in accessing (reading and up-

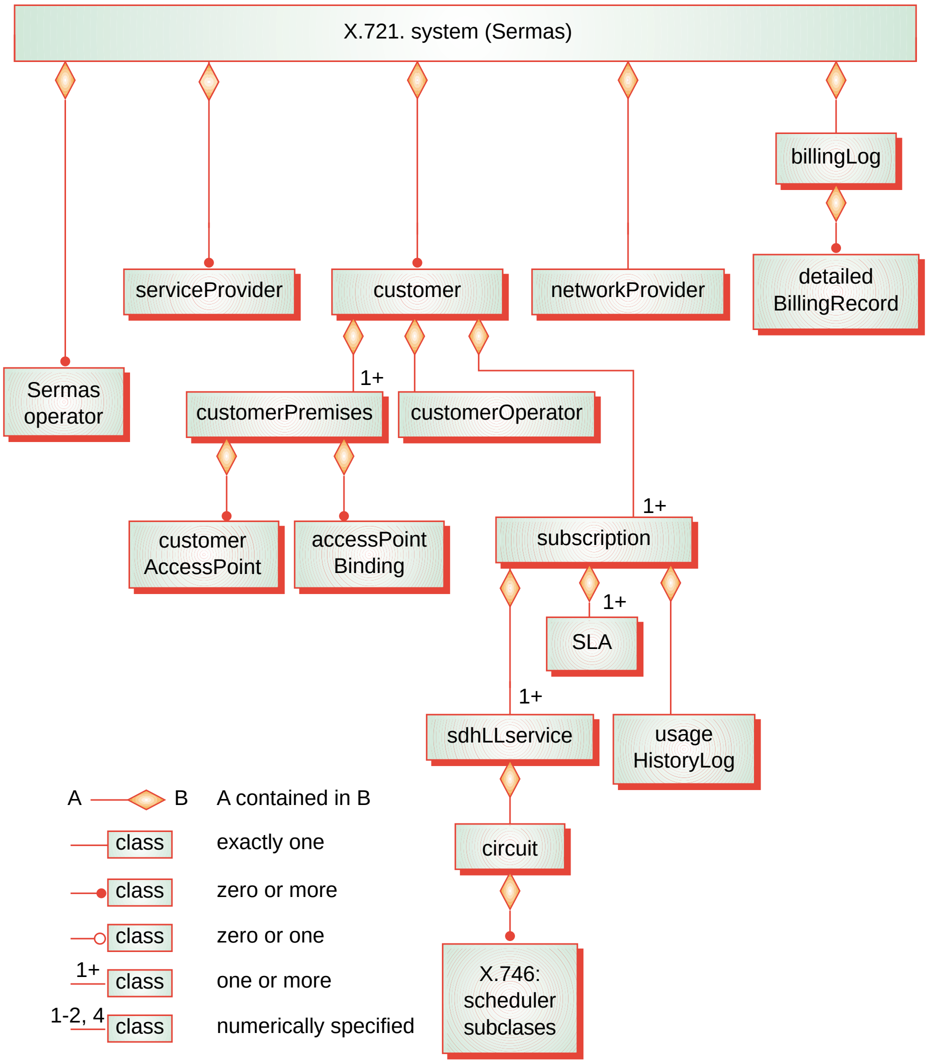


Figure 12 A simple version of Sermas, specified in OMT

dating) the Sermas server. The user application is an Applet, running in an ordinary, Java-enabled Web browser.

The application architecture

The object classes of the object models of X.721, M.3100, ETSI GOM and the actual application of these in the DSC object model form the model specific part of the Jada example system, especially the DSC part. The Sermas part builds partly on this, but introduces other kinds of model object classes, as the customer object class, which typically may be used in other systems where customers are handled. Examples of model specific object classes in KOF are Customer and Address.

The objects of the user interface applet are obvious examples of interface objects. Other examples are objects that interface Sermas with DSC and KOF.

KOF has typical model specific objects like orders and invoices (common to many Order-Invoicing systems), as well as application specific objects. Sermas has, among others, customer and customer access points as model objects, and subscriptions as application objects. Figure 12 shows an overview of Sermas' object model.

Conclusion

Jada, a prototype environment for developing management systems in Java based upon GDMO specifications, has been presented. Jada extends the basic distributed architecture of RMI objects by the architecture defined by the GDMO object model.

The basis in terms of GDMO gives Jada a benefit in domains where GDMO is already used. The example application of Jada contains a GDMO specified network model and this was readily translated to Java.

For design of applications that do not have a domain object model GDMO specification, Jada has been used

- for generating a Java design from an OMT model by specifying the OMT model in GDMO and then translating it to Java, and
- for generating a Java design from an existing database application, making the design directly in GDMO and then translating it to Java.

The benefit of using GDMO for the object model is that the documentation being produced by the Jada tool will be the same for standardised GDMO models as for the design models.

Arne Solevåg Hatlen is Research Scientist at Telenor R&D, where he has been involved in systems planning and systems planning methodology work in the area of Telecommunications Management Network (TMN). His main interests are object-orientation in design and implementation and he is currently working in a Telenor R&D project dealing with the object web and middleware technologies for the Internet.

e-mail: arne.hatlen@fou.telenor.no

Øystein Myhre is Research Scientist at Telenor R&D, where he is engaged in the evolution of the Internet. He has been involved in developing TMN systems at Telenor and at Siemens Telecom Norway. His experience with object-orientation goes back to the development of early implementations of the object-oriented language Simula at the Norwegian Computing Center and at the Norwegian Defence Research Establishment.

e-mail: oystein.myhre@fou.telenor.no

Birger Møller-Pedersen is Senior Research Scientist at NorARC, Applied Research Center, Ericsson Norway, Software Engineering.

e-mail: etobmp@eto.ericsson.se

Stig Jarle Ness is Research Scientist at Telenor R&D, where he has been employed since 1996. His interests are middleware, distributed computing and object-oriented systems.

e-mail: stig.ness@fou.telenor.no

References

1 Mathiesen, L et al. *Objektorienteret Analyse.* Forlaget Marko, 1993.

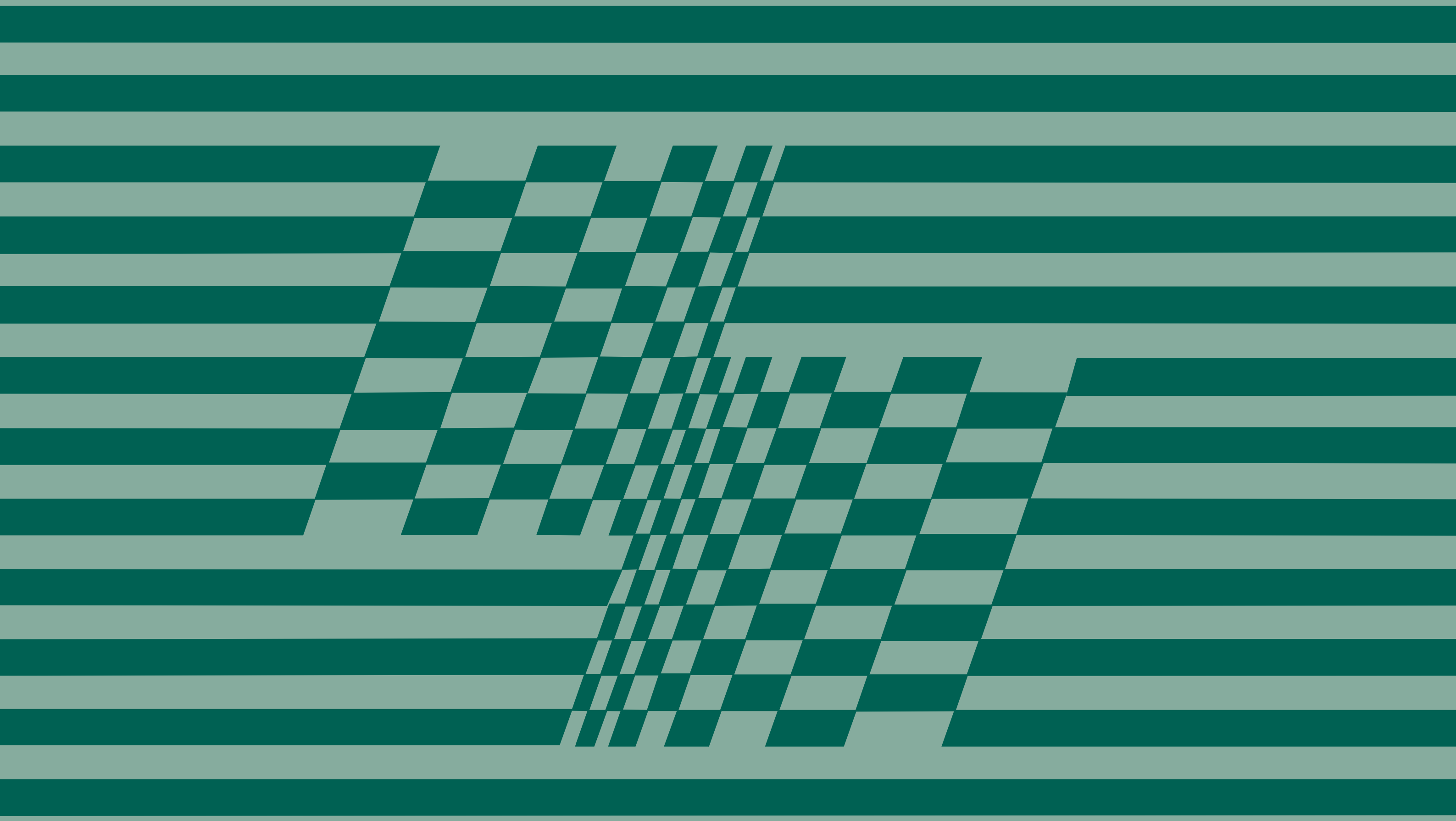
2 Møller-Pedersen, B et al. *Jada : Java-, Architecture, Development, and Application.* Kjeller, Telenor R&D, 1998. (Note FoU N 01/98.)

3 Breivik, T et al. *Input to the Sermas Requirements Specification.* Kjeller, Telenor R&D, 1996. (Note FoU N 16/96.)

4 ITU-T. *Information Technology : Open Systems Interconnection : Structure of Management Information : Definition of Management Information.* Geneva, ITU, 1992. (ITU-T Recommendation X.721.)

5 ITU-T. *Information Technology : Open Systems Interconnection : Structure of Management Information : Guidelines for the Definition of Management Objects.* Geneva, ITU, 1992. (ITU-T Recommendation X.722.)

Special



Mathematical Model and Algorithms for FABONETT/SDH

RALPH LORENTZEN

1 Introduction

The PC-based planning tool FABONETT has earlier been established as an aid for planning the deployment of service access points in PDH-based access networks (see [1]). What is described here is a related tool to be used for the same problem in SDH-based access networks. This new tool, which has been called FABONETT/SDH, has been created through a joint effort by Telenor Research and Development, Det Norske Veritas and regional network planners in Telenor Nett. FABONETT/SDH has been developed for use by network planners responsible for planning Telenor's access networks.

FABONETT/SDH is an integer programming model which attempts to find the lowest cost SDH-based access network structure which meets all capacity and connectivity requirements. The integer program is solved using a combination of a branch and cut algorithm with dynamic variable and constraint generation, and heuristics.

The planner can choose which categories of variables which will be treated as integer variables in the branch and cut phase of the algorithm, and which variables will be determined by using heuristics. The user interface allows the planner to make modifications to the solution found, and check feasibility and cost.

FABONETT/SDH has a graphical and table based user interface, and most of the development work has been directed to the implementation of this interface. In this paper, however, only the mathematical model and algorithms used in FABONETT/SDH are described.

2 The service access point structure design problem

Here a short description will be given of the service access point structure design problem to be solved.

We are given a local switch (*LS*) and a set of main distribution points (*MD*-s) together with a set of what we call special subscribers (*SS*-s). The MD-s and the SS-s can be connected directly to *LS* or via service access points (*SAP*-s) where multiplexing is done. FABONETT/SDH operates with copper cables, fibre cables and, by stretching the analogy, radio cables. An SS must be connected to *LS* or to a SAP by either fibre or radio cables. An MD must be connected to *LS* or to a SAP by copper cables. A sequence of cables which connects an MD or an SS to a SAP or to *LS*, or which connects a SAP to another SAP or to *LS*, is called a *connection.* The SAP-s may belong to SDH rings which must go through *LS*. In an SDH ring there are connections between pairs of contiguous SAP-s in the ring, and connections between *LS* and SAP-s adjacent to *LS* in the ring.

Each cable is placed in a sequence of contiguous *trace sections*. A trace section is characterized by its *cost, length, type* and one or more *section codes*. The trace section type determines inter alia which cable types can be placed in the trace section. Typical trace section types are *conduits* and *ducts* (existing or new), *trenches* of different categories, *air cable sections* and *radio sections*.

A section code is simply a positive integer. Two trace sections share a common section code if events causing damage to the two trace sections are assumed to be positively correlated. A connection inherits the section codes from the trace sections used by the cables forming the connection. Two connections belonging to the same SDH ring may not share a section code.

FABONETT/SDH operates with PDH SAP-s and SDH SAP-s. All SDH SAP-s are assumed to contain *add/drop multiplexers* (*ADM*-s). If an MD is directly connected to a SAP, the SAP must contain *RSS/RSU*. An SDH SAP can be a *Transmission Point* (*TP*). A TP does not contain RSS/RSU. Consequently, SS-s and other SAP-s, but no MD-s, can be directly connected to a TP.

FABONETT/SDH will not propose new PDH SAP-s. It may, however, propose that PDH SAP-s, which in the existing network are directly connected to *LS*, should be connected to an SDH SAP instead.

All SDH rings pass through *LS* and are either *STM1* or *STM4 SNCP* rings.

The SDH SAP-s are ordered hierarchically: SDH SAP-s which can only be connected directly to *LS* or placed in SDH rings through *LS* are called *S1 SAP*-s. SDH SAP-s which can only be connected directly to *LS* or S1 SAP-s are called *S2 SAP*-s. SDH SAP-s which can only be connected directly to *LS*, S1 SAP-s or S2 SAP-s are called *S3 SAP*-s. S1 SAP-s may belong to STM1 or STM4 SDH rings or not belong to rings at all. S1 SAP-s which belong to rings are called *ring SAP*-s. By extension of language *LS* is also denoted as an *S0 SAP*. If an S2 (S3) SAP *S* is connected to an S1 (S2) SAP *S'*, we say that *S* is *subordinate to S'*.

PDH SAP-s may be directly connected to *LS* or to SDH SAP-s. There are two categories of PDH SAP-s, namely those which require only a single connection, and those which require double connection (i.e. two connections with no section code in common) back to *LS*. The PDH SAP-s will normally be connected to *LS* in a way specified by the user. If this is not done, a PDH SAP which requires double connection to *LS* will be allowed to be singly connected to a ring SAP.

An MD may be directly connected through copper cables either to *LS* or to a SAP which is not a TP. A PDH SAP can only have connected to it MDs which either are connected to it in the existing network or which the planner explicitly connects to it. There may be subscribers connected to an MD who require the MD to be connected directly either to *LS*, to a ring SAP or to a PDH SAP with double connection to *LS*.

A circuit connecting an SS to *LS* belongs to one of two types, namely *regular circuits* and *singular circuits*. The singular circuits must be directly connected to *LS* through fibre. The regular circuits can be connected directly to *LS* or to a SAP through fibre or radio. Some SS-s may require that their regular circuits are connected directly either to *LS*, to a ring SAP or to a PDH SAP with double connection to *LS*. A PDH SAP can only have connected to it SS-s which are connected to it in the existing network or which the planner explicitly connects to it. By convention we say that an SS is connected to a SAP (or *LS*) if the SS's regular circuits are connected to the SAP (or *LS*).

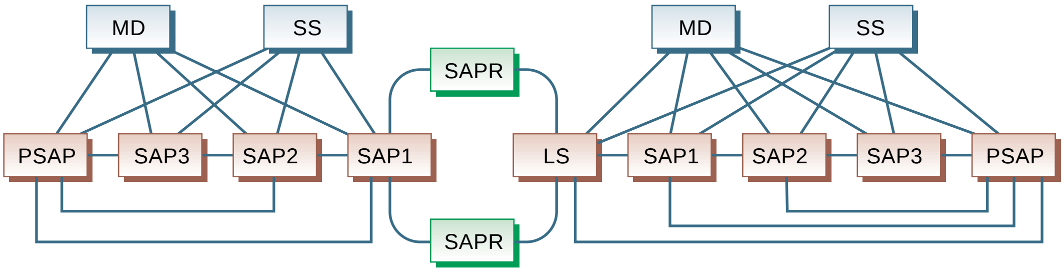


Figure 1 Schematic view of network structure

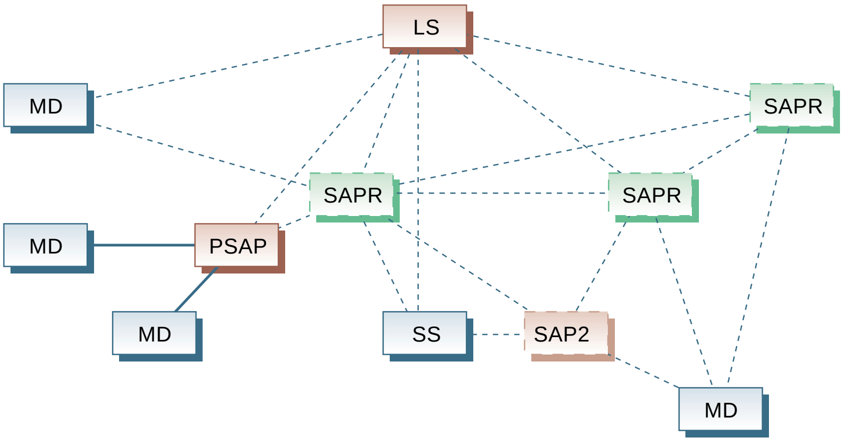


Figure 2 Input with candidate network elements

Based on the location of *LS*, locations of MD-s, SS-s, existing cables, existing and candidate trace sections, existing and candidate SAP-s, FABONETT/SDH tries to find the lowest cost network design. The design problem is formulated as an integer program which is solved by a combination of linear programming with dynamic row and column generation, branch and bound, and heuristics.

Figure 1 gives a schematic view of the network structure. Figure 2 shows an input network structure with candidate SAP-s and trace sections. Figure 3 shows the same structure together with the chosen SAP-s and connections.

FABONETT/SDH does not invent possible locations for candidate SAP-s and candidate trace sections. All candidate SAP-s and trace sections must be provided by the user.

Since FABONETT/SDH does not necessarily solve the design problem to a theoretical optimum, the planner must inspect the solution and sometimes make model reruns with slightly altered input. The planner may for example question FABONETT/SDH's selection of a particular SAP candidate and wish to make a rerun with this SAP excluded. FABONETT/SDH's input format makes this possible without erasing the SAP candidate from the input. Or the planner may question the correctness of connecting a particular SS directly to *LS*. A rerun may then be made where the planner specifies which SAP-s the SS should be allowed to connect to.

The planner has a problem of a dynamic nature. In establishing the best network structure the development of the demand structure over time must be taken into consideration. FABONETT/SDH is a static 'one shot' model. Some simple features for 'dynamic use' are, however, built into FABONETT/SDH. The planner can give selected SAP and trace section candidates the label 'preferred' and give them a bonus. Then two FABONETT/SDH runs are made. First a 'future run' is made where the circuit demands represent some future point in time. Then the main run is made where some or all the candidate SAP-s and trace sections chosen in the future run are labelled 'preferred' and given a suitable bonus.

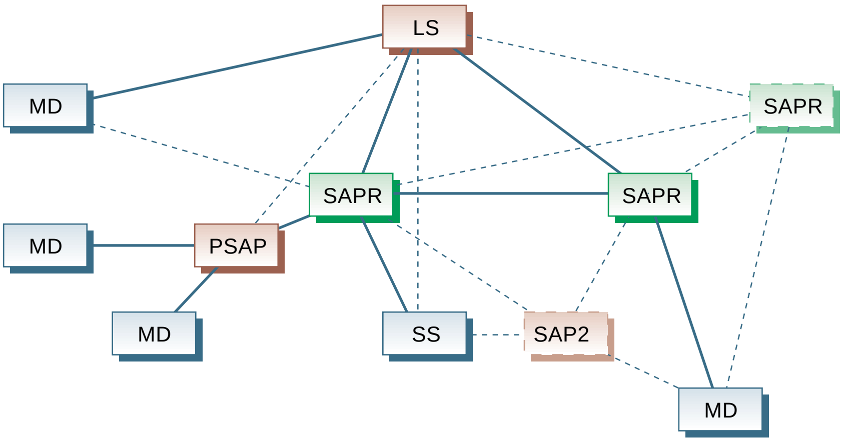


Figure 3 Resulting design

3 FABONETT/SDH input in broad terms

The complete input in the form that the planner has to present it, both tabular and graphical on background maps, is described on FABONETT/SDH's help pages. In order to give an impression of what input data are needed, an overview of main input items is given below. Not all of these data items are used in the integer programming model, and it will not be apparent how the individual data elements relate to the model. In the next section a detailed description will be given of the input which is directly related to the formulation of the integer program.

- Local switch:
  - location
  - cost per line directly from MD-s
- Candidate service access points:
  - location
  - status ('existing', 'free', 'required', 'excluded', 'preferred')
  - type (single/double connection PDH SAP, SDH SAP, ring SAP, TP, order in hierarchy)
  - capacity (maximum number of lines)
  - bonus (if status is 'preferred')
- Main distribution points:
  - location
  - SAP type requirement
  - number of subscribers of various types
  - number of 2 Mb/s circuits coming from subscribers
  - maximal distance to connecting SAP
- Special subscribers:
  - location
  - circuit requirements (nMb/s for regular circuits and fibre pairs for singular circuits)
  - which candidate SAP-s the SS can be connected to
- Nodes (end points of trace sections which are not locations of *LS*, candidate SAP-s, MD-s or SS-s):
  - location
  - cable splicing ('possible', 'not possible')
  - signal regeneration ('existing', 'possible', 'not possible')
- Trace sections:
  - location
  - length
  - type
  - status ('existing', 'free', 'compulsory', 'excluded', 'preferred')
  - bonus (if status is 'preferred')
  - section codes
- Existing cables:
  - type
  - number of available pairs
  - which trace sections the cable is placed in
- Service access point types:
  - single/double connection PDH SAP, STM1/STM4 SDH SAP, ring SAP, TP
  - capacity (maximum number of lines from MD-s)
  - life cycle cost for all components (RSS/RSU, connections, cabinet, power etc.)
- Subscriber types:
  - traffic in erlangs
- Cable types:
  - medium (copper, fibre, radio)
  - number of pairs
  - cost per metre
  - economic life time
  - trace section types the cable type may be placed in
- Trace section types
  - cost per metre

4 The integer programming model

4.1 General

Here we describe the core of FABONETT/SDH, namely the integer programming model. First we introduce some more notation. Then we describe the input which is used to formulate the constraints. Finally, the variables and constraints are described, and the form of the cost function is outlined.

We set up an undirected network which we call the *link network* with two types of links, namely *trace section links* and *cable links*. The cable links represent pieces of existing cables between contiguous nodes on the cable where cable splicing may take place. The trace section links represent trace sections (existing or not) where new cables may be placed. A trace section link inherits the section codes from the trace section it represents. A *path* in the link network is a sequence of contiguous links. All trace section links without section codes (usually ducts) are duplicated, and the original and the duplicate link receive different section codes. The duplicate links are given cost 0, but 'pushes' on its original as indicated in the constraints (21) in section 4.4 below.

As indicated earlier we operate with copper cables, fibre cables and, by analogy, 'radio cables'. A radio cable contains one pair only. Its capacity will typically be 155 Mb/s or 622 Mb/s. The regular demands of SS-s can be routed on radio cable pairs with lower capacities, e.g. 34 Mb/s or 8 Mb/s.

4.2 Notation

In order to describe the integer programming model we introduce some notation.

G geographical node containing candidate SAP-s

S service access point

U special subscriber

M main distribution point

R SDH ring

l cable or trace section link

r(U) no. of fibre/radio pairs carrying regular circuits to SS *U* (normally 0 or 1)

s(U) no. of fibre/radio pairs carrying singular circuits to SS *U*

P(M) required no. of copper pairs connected to MD *M*

TO1(M) no. of 2 Mb/s from MD *M* which do not require protection in ring

TO2(M) no. of 2 Mb/s from MD *M* which do require protection in ring

TO1(U) for SS *U*'s regular demand, no. of 2 Mb/s which do not require protection in ring

TO2(U) for SS *U*'s regular demand, no. of 2 Mb/s which do require protection in ring

TO1(S) no. of 2 Mb/s from PDH SAP *S* which do not require protection in ring

TO2(S) no. of 2 Mb/s from PDH SAP *S* which do require protection in ring

TOCAP(R) capacity for 2 Mb/s in ring *R* (126 for STM-1-rings and 504 for STM-4-rings)

LIN(M) no. of lines from MD *M*

LINCAP(S) maximum no. of lines to SAP *S*

Γ(S) a minimal set of MD-s which can be connected to SAP *S* and which together exceed the line capacity of SAP *S*

f_l no. of available fibre/radio pairs in cable link *l*

k_l no. of available copper pairs in cable link *l*

p_{fl} an upper bound on the number of SAP-s and SS-s which need fibre/radio pairs in trace link *l*

p_{kl} an upper bound on the sum of fractions of MD-s copper pair requirements which can pass through link *l*

TRAF1(M) traffic (in erlangs) on lines from MD *M* which do not require protection in ring

TRAF2(M) traffic (in erlangs) on lines from MD *M* which require protection in ring

E_0 and *E* the constant and rate of increase of the linearized erlang function expressing, for a given blocking probability, the number of channels needed as a function of traffic

4.3 Variables

When we below state the condition for a variable to be equal to 1 it is assumed that the variable is 0 otherwise.

$x_S = 1$ if SAP *S* is established

x_{MS} fraction of traffic and circuits from MD *M* which goes to SAP *S* (*S* may be *LS*)

$x_{US} = 1$ if the regular circuits from SS *U* go to SAP *S* (*S* may be *LS*)

$x_{SS'} = 1$ if SAP S is subordinate to SAP *S'* (*S'* may be *LS*)

$x_R = 1$ if STM ring *R* is established

$x_i = 1$ if a radio mast is established in node *i*

$x_l = 1$ if trace link *l* is established

$x_{ld} = 1$ if the duplicate trace link to trace link *l* is established

$x_{2SS'}$ no. of 2 Mb/s in an STM1 connection between the SDH SAP-s *S* and *S'* where *S* is subordinate to *S'*

x_{2SR} no. of 2 Mb/s (stand-by circuits included) to ring SAP *S* routed in ring *R*

x_{MGv} no. of copper pairs from MD *M* to *LS*/SAP node *G* via path *v*

x_{UGv} no. of fibre/radio pairs from SS *U* to *LS*/SAP node *G* via path *v* carrying regular circuits (0 or 1)

x_{Uv} no. of fibre pairs from SS *U* to *LS* via path *v* carrying singular circuits

x_{SGv} no. of fibre/radio pairs from SAP *S* to LS/SAP node *G* via path *v* (0 or 1)

y_{MG} number of lacking paths from MD *M* to node *G*

y_{UG} number of lacking paths for regular circuits from SS *U* to node *G*

$y_M = 1$ if there is no connection from MD *M*

y^s_U number of singular circuits to SS *U* which lack connection

$y^r_U = 1$ if the regular circuits to SS *U* lack connection

$y_G = 1$ if no SAP is established in node *G*

y_{fl} extent of fibre/radio capacity violation in trace link *l*

y_{kl} extent of copper capacity violation in trace link *l*

The *y* variables are introduced in order to avoid infeasibilities when solving the integer program. They are given large coefficients in the objective function.

4.4 Constraints

$$\sum_v x_{MGv} - P(M) \sum_{M \in G} x_{MS} + y_{MG} \geq 0 \tag{1}$$

$$\sum_v x_{UGv} - r(U) \sum_{S \in G} x_{US} + y_{UG} \geq 0 \tag{2}$$

$$\sum_v x_{Uv} + y_U^s \geq s(U) \tag{3}$$

$$\sum_S x_{MS} + y_M = 1 \tag{4}$$

$$\sum_S x_{US} + y_U^r = 1 \tag{5}$$

$$x_S - x_{MS} \geq 0 \tag{6}$$

$$x_S - x_{US} \geq 0 \tag{7}$$

$$-\sum_M LIN(M) x_{MS} + LINCAP(S) x_S \geq 0 \tag{8}$$

(generated only if left hand side can be < 0)

$$\left(\left|\Gamma(S)\right| - 1\right) x_S - \sum_{M \in \Gamma(S)} x_{MS} \geq 0 \tag{8'}$$

for all minimal Γ(S) (8') is an alternative to (8)

$$63 x_{SS'} - x_{2SS'} \geq 0 \tag{9}$$

(not generated if *S* is a PDH SAP)

$$x_{S'} - x_{SS'} \geq 0 \tag{10}$$

$$x_S - \sum_{S'} x_{SS'} = 0 \tag{11}$$

$$\sum_v x_{SGv} - \sum_{S' \in G} x_{SS'} \geq 0 \tag{12}$$

$$TOCAP(R) x_R - \sum_S x_{2SR} \geq 0 \tag{13}$$

$$x_S - \sum_{R \ni S} x_R = 0 \tag{14}$$

$$\begin{aligned} &\sum_{S'} x_{2SS'} - \sum_{S' \notin PDH} x_{2S'S} - \sum_{S' \in PDH} TO1(S') x_{S'S} - \sum_U TO1(U) x_{US} \\ &-\left[\frac{E_0 x_S}{30} + \sum_M \left(\frac{E \cdot TRAF1(M)}{30} + TO1(M)\right) x_{MS}\right] = 0 \end{aligned} \tag{15}$$

(*S* being a non-ring SAP)

$$\begin{aligned} &\sum_{R \ni S} x_{2SR} - \sum_{S' \notin PDH} x_{2S'S} - \sum_{S' \in PDH} \left[TO1(S') + 2TO2(S')\right] x_{S'S} \\ &-\sum_U \left[TO1(U) + 2TO2(U)\right] x_{US} \\ &-\left[\frac{E_0 x_S}{30} + \sum_M \left(\frac{E \cdot TRAF1(M)}{30} + TO1(M)\right) x_{MS}\right] \\ &-2\left[\frac{E_0 x_S}{30} + \sum_M \left(\frac{E \cdot TRAF2(M)}{30} + TO2(M)\right) x_{MS}\right] = 0 \end{aligned} \tag{16}$$

(*S* being a ring SAP)

$$-\sum x_{SGv} - \sum x_{UGv} - \sum x_{Uv} - \sum x_R \geq -f_l \tag{17}$$

where the sums are over paths and rings which pass through cable link *l*.

$$-\sum x_{SGv} - \sum \frac{x_{UGv}}{r(U)} - \sum \frac{x_{Uv}}{s(U)} - \sum x_R + p_{fl} x_l + p_{fl} y_l \geq 0 \tag{18}$$

where the sums are over those paths and rings which pass through trace link *l*.

$$-\sum_{MGv} x_{MGv} \geq -k_l \tag{19}$$

where the sum is over those paths which pass through cable link *l*.

$$-\sum_{MGv} \frac{x_{MGv}}{P(M)} + p_{kl} x_l + p_{kl} y_{kl} \geq 0 \tag{20}$$

where the sum is over those paths which pass through trace link *l*.

(21) $x_l - x_{ld} \geq 0$

(22) $x_i - x_l \geq 0$ if node i is one of the end points of link l.

(23) $$\sum_{S \in G} x_T + y_G = 1$$

(1) expresses that all copper pairs from MD *M* to node *G* must follow a path from *M* to *G*.

(2) expresses that a fibre pair carrying regular circuits from SS *U* to node *G* must follow a path from *U* to *G*.

(3) expresses that fibre pairs carrying singular circuits from SS *U* to *LS* must follow paths from *U* to *LS*.

(4) expresses that MD *M* must be connected to a SAP (or to *LS*).

(5) expresses that an SS *U* having regular circuits must be connected to a SAP (or to *LS*).

(6) expresses that if MD *M* is connected to SAP *S*, then *S* must be established.

(7) expresses that if SS *U* is connected to SAP *S*, then *S* must be established.

(8) or (8') express that the number of subscriber lines connected to SAP *S* must not exceed the capacity of *S*.

(9) expresses that the number of 2 Mb/s between SAP *S* and SAP *S'* cannot exceed 63.

(10) expresses that if SAP *S* is established and is subordinate to SAP *S'*, then *S'* must be established.

(11) expresses that if SAP-2 or SAP-3 candidate *S* is established, then it must be subordinate to some SAP *S'*.

(12) expresses that if SAP *S* is established and is subordinate to some SAP in geographical node *G*, then there must be a path from *S* to *G*.

(13) expresses that the number of 2 Mb/s in ring *R* must not exceed ring capacity.

(14) expresses that ring SAP *S* is established if and only if it belongs to an established ring.

(15) and (16) express flow conservation of 2 Mb/s circuits passing through SAP *S*.

(17) and (18) express that there must be sufficient number of fibre pairs in link *l* to support all fibre requirements in it.

(19) and (20) express that there must be sufficient number of copper pairs in link *l* to support all copper requirements in it.

(21) expresses that new cables can be placed in a duplicate link only if new cables are placed in the corresponding original link.

(22) expresses that if radio links are established, then corresponding radio masts must be established.

(23) expresses that at most one SAP can be established in a geographical node.

Constraints of type (13) and many variables will be generated dynamically during the solution of the problem. In addition, trace cut constraints, to be described later, will be generated dynamically.

4.5 The cost function

The cost function to be minimized has the following form:

$$\begin{aligned}&\sum c_{MGv}x_{MGv} + \sum c_{UGv}x_{UGv} + \sum c_{Uv}x_{Uv} + \sum c_{SGv}x_{SGv} + \sum c_R x_R \\ &+ \sum c_S x_S + \sum c_{SS'}x_{SS'} + \sum c_i x_i + \sum c_l x_l + \sum c_{MS}x_{MS} + \sum c_{US}x_{US} \\ (24)\quad &+ \sum c_{fl}y_{fl} + \sum c_{kl}y_{kl} + C\sum\left(y_M + y_{MG} + y_U^r + y_U^s\right).\end{aligned}$$

Here the c-s represent cost coefficients derived from cost data given in FABONETT/SDH's input tables, and *C* is a large positive constant. This cost function is detailed elsewhere. We only mention here that the coefficients c_{fl} and c_{kl} are chosen a little larger than the cost per pair for new fibre pairs and copper pairs.

5 Solution Method

5.1 General

The solution method is a combination of:

- linear programming with dynamic path generation
- branch and cut
- cost adjustment heuristics (optional)
- variable fixing heuristics.

In the branch and cut phase the variables x_S, x_{MS}, x_{US} and, optionally, x_l are required to be integer. For large problems the branch and cut phase may take too long if the x_l variables are required to be integer. Therefore the planner has the option of not requiring these variables to be integer in this phase. Their values will then, together with other variables not required to be integer in the branch and cut phase, be determined by heuristics.

5.2 Solving the linear programming relaxation with dynamic path and ring generation

We repeatedly solve linear programming relaxations of the problem. The number of path and ring variables x_{MGv}, x_{UGv}, x_{Uv}, x_{SGv}, x_R and x_{2SR} is so large that it is unrealistic to include them all in the model from the outset. The relevant path and ring variables are therefore generated dynamically during the solution process using classical column generation techniques.

The following notation is used for the shadow prices associated with the constraints which involve path and ring variables:

Constraint no.	*Shadow price*
(1)	π_{MG}
(2)	π_{UG}
(3)	π_U

(12)	π_{SG}
(13)	π_R
(14)	π_S
(16)	π_S^{bal}
(17)	π_{fl}^{ca}
(18)	π_{fl}^{tr}
(19)	π_{kl}^{ca}
(20)	π_{kl}^{tr}

Here π_R is not known for rings which are not generated.

We assume that we have a feasible basis for our system. The reduced costs for the path and ring variables relative to this basis can be expressed as follows:

	Variable	*Reduced cost*
(25)	x_{MGv}	$c_{MGv} - \pi_{MG} + \sum_{l \in v} \pi_{kl}^{ca} + \sum_{l \in v} \pi_{kl}^{tr} / P(M)$
(26)	x_{UGv}	$c_{UGv} - \pi_{UG} + \sum_{l \in v} \pi_{fl}^{ca} + \sum_{l \in v} \pi_{fl}^{tr} / r(U)$
(27)	x_{Uv}	$c_{Uv} - \pi_{U} + \sum_{l \in v} \pi_{fl}^{ca} + \sum_{l \in v} \pi_{fl}^{tr} / s(U)$
(28)	x_{SGv}	$c_{SGv} - \pi_{SG} + \sum_{l \in v} \pi_{fl}^{ca} + \sum_{l \in v} \pi_{fl}^{tr}$
(29)	x_R	$c_R - TOCAP(R)\pi_R + \sum_{l \in R} \pi_{fl}^{ca} + \sum_{l \in R} \pi_{fl}^{tr} + \sum_{S \in R} \pi_S$
(30)	x_{2SR}	$\pi_R - \pi_S^{bal}$

5.3 Reduced costs for ring variables which are not generated

In order to establish reduced costs relative to our basis for ring variable x_{2SR} which are not generated we need the values of the shadow prices π_R associated with the corresponding constraints (13) which are not generated. We shall show how we establish π_R. We pretend that (13) is included in the system and that x_R is in our basis with value 0. Since x_R is basic we have that

$$c_R - TOCAP(R)\pi_R + \sum_{l \in R} \pi_{fl}^{ca} + \sum_{l \in R} \pi_{fl}^{tr} + \sum_{S \in R} \pi_S = 0 \tag{31}$$

so that

$$\pi_R = \left(c_R + \sum_{l \in R} \pi_{fl}^{ca} + \sum_{l \in R} \pi_{fl}^{tr} + \sum_{S \in R} \pi_S \right) / TOCAP(R). \tag{32}$$

5.4 Finding not already generated path variables with minimum reduced cost

From our cost model (which is not described here) we know that the costs for the path variables x_{MGv} , x_{UGv} , x_{Uv} and x_{SGv} can be decomposed in a series of link terms. Therefore the problem of finding a variable for each path variable type which has minimum reduced cost becomes a shortest path problem in the link network where the length of a link is the sum of the relevant π term (i.e. one of the terms π_{kl}^{ca} , π_{fl}^{ca} , $\pi_{kl}^{tr} / P(M)$, π_{fl}^{tr} , $\pi_{fl}^{tr} / r(U)$, $\pi_{fl}^{tr} / s(U)$) and the link term in the cost decomposition.

For each variable type we shall include in the linear program a variable with minimum reduced cost if this reduced cost is negative.

5.5 Finding not already generated ring variables with minimum reduced cost

When we generate a new ring R we shall always at the same time generate the ring variables x_R and x_{2SR} belonging to the ring together with the corresponding constraint (13).

In section 5.4 we have seen that the variables x_R can be assumed to be basic for new rings so that these variables need not be priced out. When we shall establish ring variables of the type x_{2SR} associated with a SAP S in a geographic node G with minimum reduced cost, we see that we need to find a ring through S and LS which minimizes

$$c_R + \sum_{l \in R} \pi_{fl}^{ca} + \sum_{l \in R} \pi_{fl}^{tr} + \sum_{S \in R} \pi_S . \tag{33}$$

From the cost model we know that the cost coefficient c_R can be decomposed in a term not associated with links and a series of link terms. The link independent term represents costs in *LS*. We add one half of this cost term to the link term for all links into *LS*. Therefore the problem of finding a variable of type x_{2SR} with minimum reduced cost is reduced to a shortest ring problem where the length of a link is the sum of the π associated with the link (i.e. either π_{fl}^{ca} or π_{fl}^{tr}) and the link term from the adjusted decomposition described above, and where the 'length' of SAP candidate S is given by π_S. We see that it is not remunerative to include in the ring a SAP candidate with non-negative π_S even if the ring happens to go through the corresponding geographical node.

Ideally, we should for each SAP candidate and each ring variable type introduce into the linear program a variable with minimum reduced cost if this reduced cost is negative, i.e. if the ring length is less than $TOCAP(R)\pi_S^{bal}$. Normally, we shall do exactly that, but first we will check whether with simpler means we can establish new x_{2SR} variables associated with modifications of existing rings which price out negatively. We proceed as follows:

We go through the (unmodified) rings we have generated already one by one, and for each of these rings we check whether there are ring SAP candidates which

- do not belong to the ring
- are situated in geographical nodes which the ring passes through

- do not have a link in each of the two paths to *LS* in the ring with common section code.

For each ring where this is the case we sort such SAP candidates by descending value of $-\pi_S + TOCAP(R)\pi_S^{bal}$ and classify them as untreated. Then we modify the ring by including the first untreated SAP candidate *S* and also including other ring SAP candidates located in geographical nodes which the ring passes through and which have negative π_S. Furthermore, we remove from the ring SAP candidates with positive π_S. We denote the modified ring by *R'*. If the ring length of *R'* is less than $TOCAP(R)\pi_S^{bal}$, $x_{2SR'}$ prices out negatively. If that is the case we include *R'* with all its associated variables in the linear program.

SAP candidate *S*, together with the other SAP candidates which were included in *R'*, are classified as treated, and we proceed to the next untreated SAP candidate.

If we by carrying out the above manage to identify some x_{2SR} variables which price out negatively, we introduce them, together with the corresponding ring variables into the linear program and reoptimize.

It is only when we do not manage to introduce modified rings in this way that we try to establish shortest rings through the ring SAP candidates. In that case we start again by sorting the ring SAP candidates according to descending value of $-\pi_S + TOCAP(R)\pi_S^{bal}$ and classify them as untreated. Then we start with the first untreated ring SAP candidate and try to find a shortest ring containing it. The SAP candidate is then classified as treated. If the corresponding x_{2SR} variable prices out negatively, the ring is introduced into the LP together with associated variables, and all SAP candidates belonging to the ring are classified as treated.

In the next section we shall describe an algorithm for establishing a shortest ring. This algorithm requires that all link lengths are non-negative. We have therefore a methodological problem because one or more π_S-s may be negative. The equality sign in (14) may in principle be replaced by ≥ which would imply $\pi_S \geq 0$. We have, however, used equality in order to tighten the LP relaxation.

The link network, which was originally defined to be an undirected network, is converted to a directed network by duplicating the links and giving the two duplicates opposite direction with the same length. If the 'length' π_S of ring SAP candidate *S* is negative, we can make use of this by reducing the length on *S*'s ingoing and outgoing links. This procedure can be generalized. We start with a ring SAP candidate S_0 with most negative π_S and let it form a node set *N*. We then try to successively extend *N*.

Assume there is a maximal node set *N* with the following two properties:

- All nodes in *N* can be connected through paths of length 0
- There are SAP-s in *N* with negative π_S.

We let N^c denote the complement of *N*. Let the shortest links between *N* and N^c have length $m_N > 0$, and set

$$\pi_N = \sum_{\{S \in N; \pi_S < 0\}} \pi_S .$$

We put $\mu_N = \min(m_N, -\pi_N/2)$, subtract μ_N from the lengths of all links between *N* and N^c and distribute $2\mu_N$ to the SAP-s with negative π_S in such a way that all these SAP-s still have $\pi_S \leq 0$. Then we extend *N* by including into *N* all nodes which now can be reached from nodes in *N* through paths of length 0. Then we see again if there are node sets with the two properties mentioned above and, if so, repeat the procedure.

The links in *N* which in this way have got their lengths reduced to 0 constitute a tree structure. We replace all nodes in *N*, except the node containing S_0, by two nodes, namely an *in-node* and an *out-node*. All links belonging to the tree structure which go in the direction of S_0 are changed to go between in-nodes whilst all links belonging to the tree structure which do not go in the direction of S_0 are changed to go between out-nodes. All remaining links going to the original nodes are changed to enter the corresponding in-nodes whilst remaining links emanating from the original nodes are changed to emanate from the corresponding out-node. In this way we increase the probability that shortest rings will pick up ring SAP candidates with negative π_S.

As mentioned above, SAP candidates with non-negative π_S will not be considered for inclusion in new rings. They will not cause any link length adjustment, and the nodes they are located in will in this context be considered as nodes without a SAP candidate.

5.6 Shortest ring algorithm

We shall here describe an algorithm for establishing a shortest ring passing through *LS* and a geographical node *G*.

There exists an algorithm which for several nodes *G* simultaneously establishes rings where each ring is a shortest ring passing through *LS* and one of the *G*-s. This algorithm requires, however, extensive administration and overhead. Since we operate with a relatively small number of geographical nodes containing SAP candidates, we believe it to be more efficient to apply, for each *G*, a simpler algorithm which establishes a shortest ring through *LS* and *G*. We shall now describe this algorithm which is nothing but a specialization of Busacker and Gowen's classical algorithm for finding minimal cost flows in directed networks.

We shall initially assume that all links have different section codes. Afterwards we shall describe how we proceed without this assumption. The algorithm is based on an implementation of Dijkstra's shortest path algorithm which accepts parallel links between pairs of nodes.

1. We replace all undirected links by one link in each direction, both with length equal to the length of the undirected link.
2. We apply Dijkstra's algorithm for finding a shortest path from *LS* to *G*. Let this path have length *L*. During this process we partition the nodes into two categories *K* and K^c where *K* consists of the nodes to which we have found a shortest path, and K^c consists of the remaining nodes. For *j* in *K* we let L_j be the shortest path from *LS* to *j*.

3. We give node j in K the node number L_j and the nodes in K^c the node number L. Then we modify the link length for all links l by adding i's node number and subtracting j's node number. The modified link lengths will thus satisfy the following two conditions:
   - they are non-negative,
   - they are 0 along the shortest path.
4. We delete the links which constitute the shortest path to G which we have found and assign length 0 to the corresponding links heading in the opposite direction.
5. We then apply Dijkstra's algorithm to find a shortest path from LS to G in the modified network.
6. We combine the two shortest paths we have found to make a ring by dropping all links common to the two paths.

Of course, there may not exist any ring passing through LS and G. Then we set the ring length to ∞.

Now we return to the situation where several links can share a common section code. We have three options for the solution of this problem based on one exact and two heuristic methods.

The exact method consists of using a depth-first tree search where we successively eliminate links which violate the section code requirements. We form a directed tree structure of shortest ring problems where the root node is the shortest ring problem without section codes. The tree in which we perform the depth-first search is such that the successors to a problem node are nodes representing problems where we exclude exactly one link for which the section code requirement is violated. (We remark that it is permitted to pass through links sharing the same section code between two consecutive SAP candidates in the ring.) During the tree search we continually update an upper and lower bound on the shortest ring length so that we can find a shortest feasible ring without necessarily traversing the whole tree.

The first heuristic consists simply of removing in step 4 all links from the network which have common section code with links on the shortest path we found from LS to G, and which are not heading in the opposite direction of links on this shortest path. For the rest the method is identical to the exact method.

The second heuristic is the same as the first except that we always accept the first ring we get. This heuristic will give a feasible ring if all links which share a common section code are parallel.

6 Trace cut constraint generation

6.1 Trace cut generation for paths to main distribution points

We focus on a node in the branch and bound tree where we want to add trace cuts.

We denote links l which have no x_l as *free links* and define

$$z_M = \begin{cases} 1 & \text{if all paths to } M \text{ from } LS/\text{SAP consist of free links only} \\ 0 & \text{otherwise} \end{cases}$$

The z_M variables are given a small negative cost $-\varepsilon$.

The following inequalities are included in the LP from the outset:

$$(34) \qquad \sum_{G,v \text{ in free links only}} x_{MGv} - P(M) z_M \geq 0$$

with shadow price $K_M{}^{\varepsilon}$.

For each M with $z_M < 1$ we do the following:

For the link network we assign capacity c_l to link l where c_l is set to:

$$c_l = \begin{cases} x_l & \text{if } l \text{ is not a free link} \\ \min\{P(M), \text{ the slack in } (19)\} & \text{if } l \text{ is a free link containing a copper cable} \\ 0 & \text{otherwise} \end{cases}$$

Links with $x_l = 0$ are removed from the link network.

In addition we introduce links with capacity $P(M)$ between LS/SAP candidates for M and a supernode. Then we find maximum flow between M and the supernode, with a min cut *(C,C')* where $M \in C$, and check whether

$$(35) \qquad z_M + \sum_{l \in (C,C')} x_l - \sum_{S \in C'} x_{MS} \geq 0.$$

If not, and if in addition there exists at least one link l in *(C,C')* with $x_l > 0$, we add the constraint (35) to the LP and optimize with path and ring generation.

Here we of course generate paths from the M-s for which we have added constraints (35). Such paths which only use free links get a bonus equal to the shadow price $\kappa_M{}^0$ which (34) would have got if z_M had cost 0. We shall now determine $\kappa_M{}^0$:

Assume first that z_M is basic both if z_M has cost $-\varepsilon$ and if z_M has cost 0. Then:

(36) $-P(M)\kappa_M{}^0 + \textit{remaining terms} = 0$

(37) $-P(M)\kappa_M{}^{\varepsilon} + \textit{remaining terms} = -\varepsilon.$

The terms denoted as *remaining terms* are non-negative. We assume that the basis is the same whether z_M has a positive cost or not. Then *remaining terms* are the same in (26) and (27). This gives:

(38) $\kappa_M{}^0 = \kappa_M{}^{\varepsilon} - \varepsilon / P(M).$

Assume now that z_M is non-basic (this can happen only if z_M has cost 0). Then $\kappa_M{}^0 = 0$ since (24) is satisfied with strict inequality. (26) then implies that *remaining terms* = 0, in other words that the other constraints in which z_M occurs are not binding. Then they will not be binding with a small negative cost on z_M either, so that *remaining terms* = 0 also in (27). With cost $-\varepsilon$ on z_M, z_M will always be basic so that (27) is satisfied, and we have that:

(39) $\kappa_M{}^{\varepsilon} = \varepsilon / P(M).$

This means that (38) is still valid.

If we have generated (35) for an MD M and $\kappa_M{}^0 > 0$, and the shortest path does not consist of free links only, we must generate another shortest path which uses free links only and give the resulting path (if it exists) a bonus $\kappa_M{}^0$.

6.2 Trace cut generation for rings

Again we focus on a node in the branch and bound tree where we want to add trace cuts.

We define:

F(S): the set of rings through *S* which use free links only for at least one of the paths from *S* to *LS*

FF: the set of rings which use free links only

$$w_S = \begin{cases} 1 \text{ if ring SAP } S \text{ is established,} \\ \quad \text{and the ring through } S \text{ belongs to } F(S) \\ 0 \text{ otherwise} \end{cases}$$

$$z_S = \begin{cases} 1 \text{ if ring SAP } S \text{ is established,} \\ \quad \text{and the ring through } S \text{ belongs to } FF \\ 0 \text{ otherwise} \end{cases}$$

We observe that $w_S \ge z_S$. The w_S and z_S variables are given a small negative cost $-\varepsilon$.

The following inequalities are included in the LP from the outset:

(40) $$\sum_{R\ni S, R\in F(S)} x_R - w_S \ge 0 \text{ with shadow price } \kappa_S^{F\varepsilon}.$$

(41) $$\sum_{R\ni S, R\in FF} x_R - z_S \ge 0 \text{ with shadow price } \kappa_S^{FF\varepsilon}.$$

For every ring SAP *S* with $z_S < 1$ we do the following:

We consider the link network and give capacity c_l to link l where c_l is given the value x_l if l is not a free link, and the value 1 otherwise.

Then we find the maximum flow between *S* and *LS*, with min cut *(C,C')*, and check if

(42) $$w_S + z_S + \sum_{l\in(C,C')} x_l - 2x_T \ge 0$$

is satisfied. If (42) is not satisfied, we add (42) to the LP and optimize with column generation.

Finding $\kappa_S{}^{F0}$ and $\kappa_S{}^{FF0}$ from $\kappa_S{}^{F\varepsilon}$ and $\kappa_S{}^{FF\varepsilon}$ is analogous to finding $\kappa_M{}^0$ from $\kappa_M{}^\varepsilon$. We get:

(43) $\kappa_S{}^{F0} = \kappa_S{}^{F\varepsilon} - \varepsilon$

(44) $\kappa_S{}^{FF0} = \kappa_S{}^{FF\varepsilon} - \varepsilon.$

After we have generated a shortest ring through *S* again we can have three situations:

1. The shortest ring belongs to *FF*. Then the ring is given a bonus $\kappa_S{}^{FF0} + \kappa_S{}^{F0}$.
2. The shortest ring belongs to *F(S)\FF*. Then the ring is given a bonus $\kappa_S{}^{F0}$. If in addition $\kappa_S{}^{FF0} > 0$, we must also find a competing ring in *FF* which is given a bonus $\kappa_S{}^{FF0} + \kappa_S{}^{G0}$.
3. The shortest ring does not belong to *F(S)*. If $\kappa_S{}^{F0} > 0$, we must find a competing shortest ring in *F(S)* which is given a bonus $\kappa_S{}^{F0}$. If in addition this ring belongs to *FF*, it is given an additional bonus $\kappa_S{}^{FF0}$. If $\kappa_S{}^{FF} > 0$ and the ring does not belong to *FF*, we must find a competing shortest ring in *FF* which is given a bonus $\kappa_S{}^{FF0} + \kappa_S{}^{F0}$.

One problem is how we find a shortest ring in *F(S)*. We have not found any exact method for this, so we use a heuristic. In brief, the heuristic amounts to the following:

1. We establish a shortest path *v* from *LS* to *S* in the subgraph consisting of free links only.
2. Then we establish a shortest path from *LS* to *S* where we are not allowed to use links with section code common with any of the links in *v*.

We have not yet implemented trace cut generation for rings. To what extent this will be done is yet to be decided.

7 Integer programming heuristics

7.1 Introduction

The integer programming heuristic is roughly the same which was applied in [1]. Initially, we optionally use a branch and cut algorithm with dynamic column generation at each node in the branch and cut tree where the user decides which amongst the variable types x_S, x_{MS}, x_{US}, $x_{SS'}$ and x_l should be required to be integer. During the branch and cut phase the variables y_{fl} and y_{kl} are given a temporary upper bound equal to zero.

The variables of the types x_S, x_{MS}, x_{US}, $x_{SS'}$ and x_l which are not required to be integer in the branch and cut phase are, together with the path and ring variables, fixed sequentially. For variable fixing the variables x_S are given first priority, and amongst these variables the ones that are higher in the SAP hierarchy are given higher priority. Variables of type $x_{SS'}$ are given second priority, variables of types x_{MS} and x_{US} are given third priority, and variables of type x_l are given fourth priority.

The path and ring variables are generated dynamically, so they are always fixed sequentially with fifth priority.

For large problems it may demand too much resources to require the x_l variables to be integer in the branch and cut process. On the other hand, using the variable fixing heuristic for the x_l variables may imply a significant underestimation of trace link costs. We therefore apply *trace capacity adjustment*.

7.2 Trace capacity adjustment

We assume that we have arrived at a stage where all variables of the types x_S, x_{MS}, x_{US}, and $x_{SS'}$ are integer. We remove the temporary upper bounds on the variables y_{fl} and y_{kl}, run the LP, and check for every trace link l the value of x_l. If $x_l = 0$, $y_{kl} = 0$ and $y_{fl} = 0$, we set upper bounds 0 on the variables x_l, y_{kl} and y_{fl} and classify trace link l as excluded. If $0 < x_l <$ *threshold* or $y_{fl} > 0$ or $y_{kl} > 0$ (where *threshold* is a parameter between 0 and 1), we adjust p_{fl} to $\lceil p_{fl}\, x_l + p_{fl}\, y_{fl} \rceil$ if (18) is satisfied with equality, and p_{kl} to $\lceil p_{kl}\, x_l + p_{kl}\, y_{kl} \rceil$ if (20) is satisfied with equality. Then we run the LP again.

When a variable x_l is given a lower bound 1, we set $p_{kl} = p_{fl} = \infty$. Furthermore, when all x_l have become integer, all x_l which have become 1 are given lower bound 1 (accompanied by a corresponding upward adjustment of p-s), and all x_l which have become 0 are given upper bound 0. Then a new optimization is carried out.

Capacity adjustment is currently executed only once. It may, however, be repeated several times. It is also possible to execute capacity adjustment before all variables of the types x_S, x_{MS}, x_{US}, and $x_{SS'}$ have become integer, but this has not been implemented.

8 Acknowledgement

The author wishes to thank Øyvin Eriksen who has contributed many ideas during the development of the algorithms and who has programmed the algorithms for PC.

9 Reference

1 Lorentzen, R. Mathematical model and algorithms used in the access network planning tool FABONETT. *Telektronikk*, 91 (4), 135–139, 1995.

Ralph Lorentzen is Research Scientist at Telenor R&D, working with Communications Network Planning. His interests concentrate on applications of mathematical programming.

e-mail:
ralph.lorentzen@kjeller.fou.telenor.no

Status

International Research and
Standardization Activities
in Telecommunications

Editor: Per Hjalmar Lehne

Introduction

PER HJALMAR LEHNE

As a follow-up from the previous issue of *Telektronikk* (3/4.1997) where four ACTS (Advanced Communications Technologies and Services) projects were presented (CRABS, OPEN, MoMuSys and SINUS), this issue of the Status section contains articles on the ACTS program and on Telenor's participation in general. Because the ACTS programme probably is the most important scene for setting the future telecom agenda, we feel it is important to focus on both the work, the results and how it is organised.

In the first article, Dr. Rolf B. Haugen, manager of Telenor's external relations, presents the ACTS programme, how it is organised, and makes short descriptions of the projects where Telenor participates. The ACTS programme is now in its third and last phase. After the third call for proposal has been finalized some 200 projects have been launched.

The ACTS programme does not only fund specific technological projects aimed at system design and so on. The so-called *horizontal actions* are important to disseminate results and to connect the different projects together. The *InfoWin* project is described by Mr. Thorbjørn Thorbjørnsen. This project is meant to be the ACTS Information Window, allowing information flow from the projects to the outside world as well as giving the outside world an opportunity to look into what is happening in ACTS. InfoWin publishes on the web, as well as news clips, bulletins and thematic issues. It is also responsible for the *ACTS Yearbook*.

A lot of standards work and work relating to the use of standards is going on outside the large and well known organisations (e.g., ETSI[1], ITU[2] and ISO[3]). In the final and third article of this issue, Mr. Tor M. Jansen presents *FERT*, which stands for "Forum for European R&D in Telecommunications". The goal of FERT is basically to provide strategic guidance in the European telecommunications development. The role of these types of organisations and projects is significant in the cooperative process in R&D and standardisation.

Table 1 ACTS projects with Norwegian participation

Interactive Digital Multimedia Services	
CRABS	Cellular Radio Access for Broadband Services
MAESTRO	Maintenance System based on Telepresence for Remote Operators
CUSTOM TV	
VIS-à-VIS	Fitness for Purpose of Videotelephony in Face-to-Face situations
Mobility and Personal Communication Networks	
MoMuSys	Mobile Multimedia Systems
SAMBA	System for Advanced Mobile Broadband Applications
SINUS	Satellite Integration into Networks for UMTS
STORMS	Software Tools for the Optimization of Resources in Mobile Systems
High-Speed Networking	
ASICCOM	ATM Switch for Integrated Communication Computation and Monitoring
CA$HMAN	Charging and Accounting Schemes in Multi Service ATM Networks
EXPERT	Platform for Engineering Research and Trials
JAMES	Provision of European Networking Facilities (PEN)
NICE	National Hosts Interconnection Experiment
REFORM	Resource Failure and Restoration Management in ATM-based IBCN
SMASH	Storage for Multimedia Applications Systems in the Home
TRUMPET	TMN's Regulations and Multiple Providers Environment
Photonic Technologies	
BLISS	Broadband Lightwave Sources and Systems
MEPHISTO	Management of Photonic Systems and Networks
OPEN	Optical Pan-European Network
ACTUAL	Application and Control of Widely Tuneable Lasers
Quality, Security and Safety of Communication Systems and Services	
RETINA	An Industrial-Quality TINA-Compliant Realtime DPE
Horizontal Actions	
INFOWIN	Multimedia Information for National Hosts
OPTIMUM	Optimized Network Architectures for Multimedia Services
TERA	Techno-Economic Results from ACTS
INFOBRIDGE	The Bridge from ACTS to the outside world

Per Hjalmar Lehne is Research Scientist at Telenor Research & Development, Kjeller. He is working in the field of personal communications, with a special interest in antennas and radio wave propagation for land mobile communications.

e-mail:
per.lehne@fou.telenor.no

[1] *European Telecommunications Standards Institute*

[2] *International Telecommunications Union, a UN-body*

[3] *International Standardisation Organisation*

The EU Research Programme ACTS
A General Description with Focus on Telenor Participation

ROLF B. HAUGEN

1 Introduction

ACTS is an abbreviation for 'Advanced Communications Technology and Services' and is part of the 4th framework programme in EU. The total budget is about 670 MECU which corresponds to 5 % of the total framework budget. It is a direct successor to the previous RACE-programme, known to many for its pioneering work within ATM. ACTS is divided into three (time) phases, each starting with a 'call for interest'. At present (end 1997) we have just entered the last phase, which means that there is about two more years to go. During the first two phases about 150 projects have been launched, with the third call this number will increase to about 200.

ACTS was established on 27 July 1994 by a European Council decision, the objectives being *"… To develop advanced communications systems and services for economic development and social cohesion in Europe, taking account of the rapid evolution in technologies, the changing regulatory situation and opportunities for development of advanced trans-European networks and services ..."*

There has been a clear shift in technology focus from the early RACE-programmes (RACE I and RACE II) to ACTS. The objective of RACE was to bring forward necessary technology for broadband communications (ATM), whereas ACTS is oriented towards *end-users* of advanced communications. Much of ACTS research is thus linked to *trials* and *pilots* of the thus developed technologies. In this context *photonics* is an exemption; it is commonly recognized that more fundamental research is still needed within this area in order to obtain cost-effective components. Hence, ACTS supports several research projects devoted to optical components and systems.

Formal decisions to be taken in ACTS are discussed in ACTS Management Committee (AMC). In AMC all EU/EEA countries are represented by two delegates, appointed by their respective governments. In the Norwegian case, one delegate comes from the Research Council (NFR) and the other from Telenor, a rather typical combination. AMC is an advisory group for the Council, but to my experience, its professional (i.e. project-related) advice has always been followed.

One of the most important tasks of AMC is to approve new projects. The projects are, of course, evaluated on the grounds of their technical merits, which certainly give strong guidance for the final approval. More than 90 % of the projects are approved on this basis. Occasionally there are nevertheless 'political' arguments for choosing one project to another. Any thus unresolved dispute in AMC will finally go to formal voting. This is usually not in the interest of Norway (and EFTA countries) since we have no voting rights. Of the above mentioned three phases of ACTS, only the first one applied formal voting for final approval of the projects; in the succeeding phases agreements were reached based on consensus.

Although research and technological development is the main focus of ACTS, the Council has opened for certain support measures and concerted activities. These are generally known as 'horizontal actions'. Horizontal activities might be of various types like *encourage practical experimentation, help disseminate results, promote synergies, inform external interest group, assess socio- and techno-economic factors,* etc. Some projects are directly placed under the 'horizontal action' label like Infowin and Optimum, while other activities are placed in research project pertaining to other domains. One example here is the project Nice that belongs to 'high speed networks'. I will come back to these projects later.

National Host (NH) is another concept closely associated with ACTS. The very idea behind NH stems from the notion that ACTS focuses on demonstrators and piloting. Hence, the Commission sent an invitation to all the EU/EEA members to provide infrastructure and projects facilities, called National Host, for the ACTS projects to come. The rationale was that by offering such an infrastructure, the respective countries would qualify to run corresponding ACTS projects, sort of a 'tender situation' for ACTS projects. In practice, this 'tender idea' was not really launched, but one (or more) NH was nevertheless created in every participating country.

NHs are today established in more than 20 countries and support practical experimentation of advanced communications at national and European level. They are interconnected with *commercial* services like ISDN and Internet in addition to *experimental* services like ATM (through the James network). On top of this, the NHs have agreed to operate a defined set of link services, both standardized as well as leading-edge services far from maturity. These link services support all communications needs, from e-mail to multimedia mail, video conferencing to co-operative work, high-speed file transfer, multi-site conferences and meetings, etc.

The National Hosts are jointly organized in National Hosts Forum which meets regularly, plans major conferences and demonstrations, discusses common problems and makes informal agreements.

2 Call for interest

A 'call for interest' starts with an invitation to submit proposals for projects within a predefined set of tasks. The tasks are grouped into a number of areas, called *domains.* In ACTS we are talking about six domains to be further dwelt upon below. When proposals have been submitted, they are evaluated by a group of independent experts. The experts are grouped into 'panels'; each panel with responsibility of evaluating proposals within one particular domain.

The chairman of each panel presents the result of the evaluation to AMC. This is usually done through two consecutive AMC-meetings. The first one is devoted to panel presentations with possibility of (short) questions for clarifications, whereas the next one opens for more in-depth 'cross examination' of the panels. After these 'professional' discussions of the projects, AMC normally needs one or two further meetings until the whole package is approved.

The above mentioned procedure is a rather general one in EU. It might seem long and bureaucratic, but at least in case of ACTS, the secretariat in Brussels runs it very smoothly. As an example from last call: Final date for proposals was 28 September, at which time 195 proposals had arrived. Evaluation started immediately and for the next AMC-meeting, 23 Octobre, we found on our table several printed documents containing *project descriptions, evaluation reports* (2 volumes) and *executive summary* (including statistics). Not bad for a three weeks period! The final approval was done by AMC on 25 November; hence the elapse of time from reception of proposals to approval of the new projects was two months.

3 Concertation

ACTS is an integrated research programme. The overall result it achieves will be greater than the sum of the individual projects' results. Hence, quite an effort has been made to obtain synergies from projects belonging to different domains, leading to concepts like *chains* and *concertation.* A chain is a 'horizontal' grouping of projects that contain elements of similarities, no matter which domain they belong to. In certain cases the projects may also contribute to the work of another, and some projects choose to support several different chains. This leads to a need for *internal concertation* within the programme. In this context concertation may be defined as: *'the bringing together of people and their organizations/projects, to profit from each others' knowledge and experience, to coordinate or encourage the convergence of ongoing work on the most important issues, and so to build a broadly based consensus on the way to realize advanced communications in Europe'.*

Concertation within ACTS centres on three main groupings:

- Plenary Meetings of all project managers, typically 3 or 4 times per year
- Technology oriented R&D domains, meeting in parallel after a plenary and focusing on the main technical areas of the programme
- Objective Driven Chains, each supporting a defined objective and contributing to a specific result (for example *guidelines*).

Through a Concertation Steering Committee, an ongoing effort is made to keep the work of individual chains and domains aligned both inside and outside of the Programme. Sometimes this may lead to changes in timing and priority for individual actions within the projects themselves.

3.1 Chains

For organizational convenience, individual chains are grouped into five main groups:

- BA: Broadband Access Networks: Economics and Evolution
- NI: Network Level Inter-operability and Management
- SI: Global Service Integration
- GA: Generic Access to Applications (User perspective)
- XB: External: Broadening of Awareness (Programme Level).

BA addresses issues arising from different topologies of access networks, their potential for evolution, economics, integration issues, etc.

NI deals with major issues of interworking and integration, including signalling and interoperability, roaming and location functions to provide mobility in both fixed and cellular networks. The work aims to ensure end-to-end quality of service across multiple network operators and service providers in a competitive environment.

SI seeks to decouple the characteristics of services from those of the underlying networks, for example that service can be provided irrespective of access network architectures. The work will also serve to ensure continuity of services as the underlying network technologies evolve.

GA works in a similar way on the application level, to ensure that generic applications can be supported across different industry sectors and organizations, irrespective of the different platforms and terminal equipment.

XB chains are formed to address specific needs for broadening awareness at Programme level, whenever there is no clear initiative taken by any single project or chain. That is, this group of chains represents overall Programme interests.

3.2 Guidelines

Concerted efforts of Programme projects may take on many forms, as for example realization of common demonstrators. More frequently, the concerted actions will result in *guidelines.* Guidelines are concise documents targeted towards outside users. They provide the basis for consultation and systematic collaboration with related research and policy initiatives within EU, the member states and third countries, international industry groupings, standardization bodies, etc. The guidelines present various form for strategic and/or specific technical recommendations of the ACTS programme.

Guidelines will typically point to techno-economic feasibility of alternative technological solution under specific circumstances, and so contribute to a reduction of options for product and service deployment.

3.3 Horizontal projects

As mentioned earlier, there are in ACTS several projects devoted to 'horizontal actions'. Most of these will be found in the domain named 'Horizontal actions' but some are also placed within one of the other domains. We will in this section consider three projects that all have played an important role in disseminating results both inside and outside the programme:

- James (Joint ATM Experiment on European Services)
- Nice (National Host Interconnection Experiments)
- Infowin (Multimedia Information for National Hosts).

Only the last one, Infowin, belongs to the Horizontal Actions domain.

3.3.1 James

James belongs to the Networking domain and has an objective to test and evaluate new ATM-based broadband experimental services and applications throughout Europe. As a *research project* James aims to demonstrate the utility and added-value of advanced ATM networks and contains tasks like:

- ATM VP Bearer Service
- IP over ATM
- ATM Switched VC Service
- LAN Interconnection Service
- SMDS over ATM Service
- Network Management.

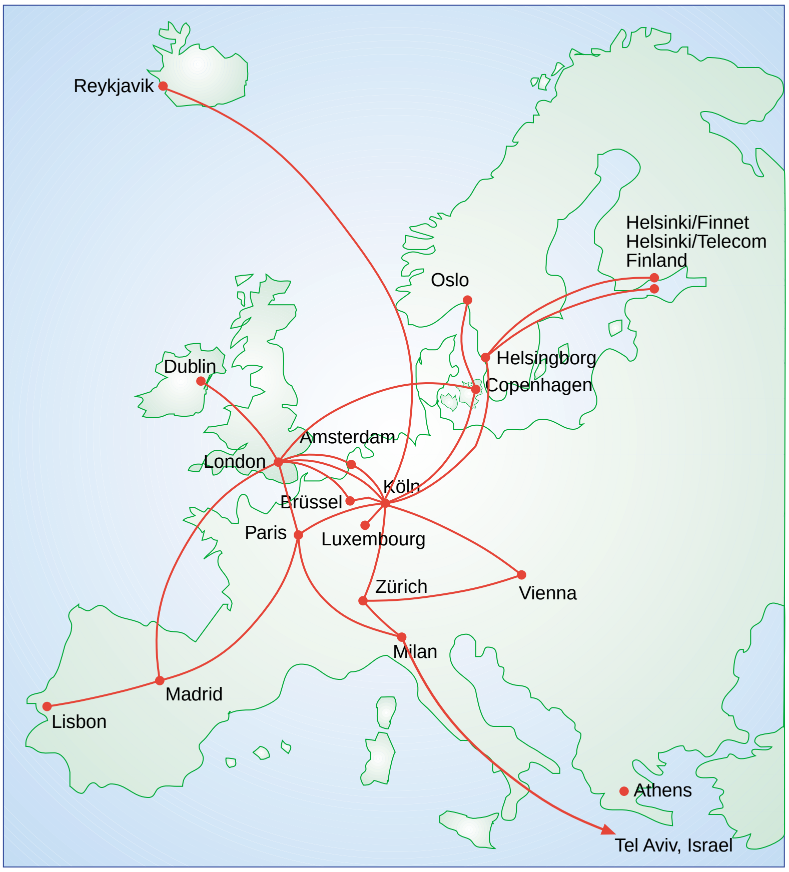


Figure 1 Points of presence of James

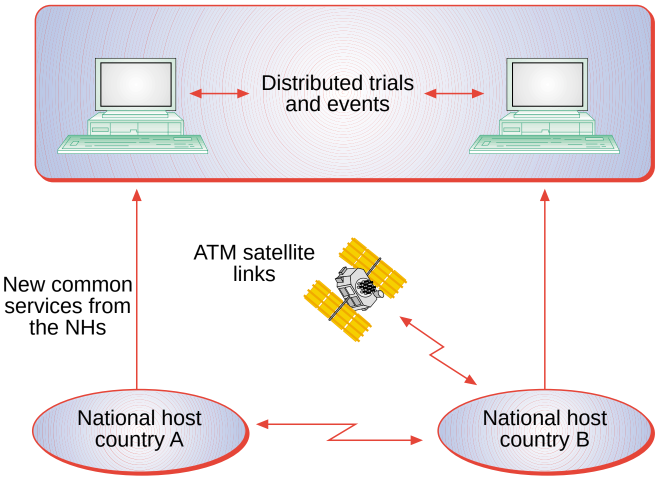


Figure 2 Nice assists the NHs in their service experiment

However, in addition to the above mentioned areas, which are research areas in their own merits, James has (had) the important function of providing ATM infrastructure to several other ACTS projects. In particular, the various National Hosts are tied together by the James network, as depicted in Figure 1. In this sense James has been a very important horizontal project within the ACTS programme. Members of the James project are the major tele-operators in Europe that, in fact, have subsidized the usage of the network.

The James projects will unfortunately be closed by end of March 1998, after which time broadband field trials in ACTS have to be based on commercial solutions.

3.3.2 Nice

The objective of Nice is to assist the National Hosts to provide common, international broadband applications on an ATM infrastructure. It belongs to the Networking Domain. The project concentrates on teleconferences and distributed meetings associated with ACTS and other EU R&D activities. The National Hosts are the prime vehicle for delivering these experimental services.

Nice is thus integrating systems so as to enable groups of NHs to provide common, international broadband teleconferencing and fast asynchronous services based on ATM. In 1996 the participation in Nice was extended to Central and Eastern Europe (CEE) and the Newly Independent States of former Soviet Union (NIS).

Examples of activities are:

- Advanced Broadband summer school 1996: 22 sites with more than 1000 people
- Distributed meeting for G7's GIBN project
- Linking Russian and Western partners in workshops.

3.3.3 Infowin

The objective of the Infowin project is to provide the ACTS Information Window. This window allows information to flow from ACTS projects to the outside world, and also helps the outside world to be visible to the ACTS projects.

Infowin provides support for the internal communications of ACTS, within the project as well as between projects and the Commission. The project is structured around information and marketing, editorial and dissemination, on-line as well as through other means. Updated information is always to be found on the Web-site: www.infowin.org/.

Type of information from Infowin is:

- *Newsclips:* Info published and distributed via WWW and e-mail twice a month. Information about ACTS projects and new technology developments.
- *Bulletins*, appearing four times a year. Cover special research topics of ACTS or other topics of current interest, important events, articles, and publications.
- *Thematic Issues* published quarterly cover special research topics of ACTS or other topics of current interest. A Thematic Issue can be a regular publication or it may be a workshop.

4 Domains

The total project *budget* of ACTS is about 630 MECU, which corresponds to NOK 5 billion. Every project is assigned to one single domain as their 'home base'. As already mentioned, there are five technology-oriented domains and a sixth housing those horizontal projects which do not belong to one of the technology oriented ones. The domains are:

• Interactive Digital Multimedia Services	(162 MECU)
• Photonic Technologies	(104 MECU)
• High Speed Networking	(75 MECU)
• Mobility and Personal Communications Networks	(115 MECU)
• Service Engineering, Security and Comm. Management (TMN)	(143 MECU)
• Horizontal Domain	(31 MECU)

The figures in brackets represent the *initial* distribution of the funding amongst the various domains. The final numbers, as of today, show a small shift in the favour of Multimedia Services and High Speed Networking.

The number of Telenor participating projects, to be discussed below, might seem a bit ambitious. But since there is a mix of projects from all three phases of ACTS, some projects (from phase 1) are about to terminate, whereas the projects from the third call are on the point of starting up.

5 Projects with Telenor participation

5.1 Domain 1: Interactive Digital Multimedia Services

The tremendous interest in real-time interactive multimedia services indicates a large potential market for these services. Hence, this domain has attracted a large amount of project proposals, many with a very high professional quality. The result has been an approval of projects that exceeds the initial budget by about 20 MECU.

The domain has been subdivided into the following three sub-domains:

SD1: Multimedia Content manipulation and management

SD2: Interactive Distribution and Transmission

SD3: Server based Multi Media Services.

Telenor participates in four projects within sub-domain SD1:

- **Momusys:** Mobile Multimedia Systems
- **Custom TV**
- **Maestro:** Maintenance System based on Telepresence for Remote Operators
- **Vis à Vis:** Fitness for Purpose of Videotelephony in Face-to-Face Situations.

Momusys started already in the first phase of ACTS and is henceforth in its third year of running. An extension (Momusys-Ext) was approved at the third call. The objective of the project is to develop and validate possible technologies for audio-visual functionalities for mobile systems. The work has been heavily based on the standardization work in ISO MPEG-4 as well as ITU and ETSI; in fact, Momusys is a provider of software programs to the MPEG work. The project has also developed an 'MPEG-4 terminal' that is used in the field trial. The project is progressing well with many promising results.

Custom TV has the main objective to develop, demonstrate and validate technology for user-friendly *screen customization* and interactive program selection in a multimedia broadcast environment. The project will demonstrate the evolutionary path from MPEG-2 to MPEG-4 and MPEG-7. Custom TV is a phase 3 project, starting end 97 / beginning 98.

Maestro aims at developing the use of *telepresence* for *maintenance, installation* and *repair* of mechanical equipment. It is particularly dedicated to the training of complex maintenance/ installation scenarios for remote users. The resulting technology should enable users to be trained by connecting to a 'Virtual Showroom' where they can learn maintenance procedures through computer-augmented video-based telepresence. The project is progressing well and the demonstrator is about to be started.

Vis à Vis is a third call project, starting at the beginning of 1998. The project aims at examining videotelephony for communication in companion with face-to-face communications.

Telenor is attending only one project in sub-domain SD2:

- **Crabs:** Cellular Radio Access for Broadband Services.

Crabs is a project of 1017 man months divided among 9 partners and 3 associated partners and with Telenor as a prime contractor.

The main objective of Crabs is to develop and demonstrate a *cellular radio system* to provide *broadband interactive* multimedia services, e.g. digital television. The system operates in the 40 GHz frequency band, and field trials will take place in five European countries: Czech Republic, Greece, Italy, UK and Norway.

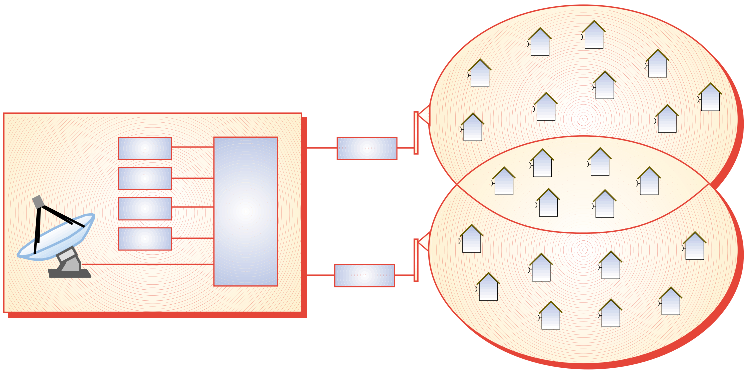


Figure 3 The cellular broadband network of Crabs

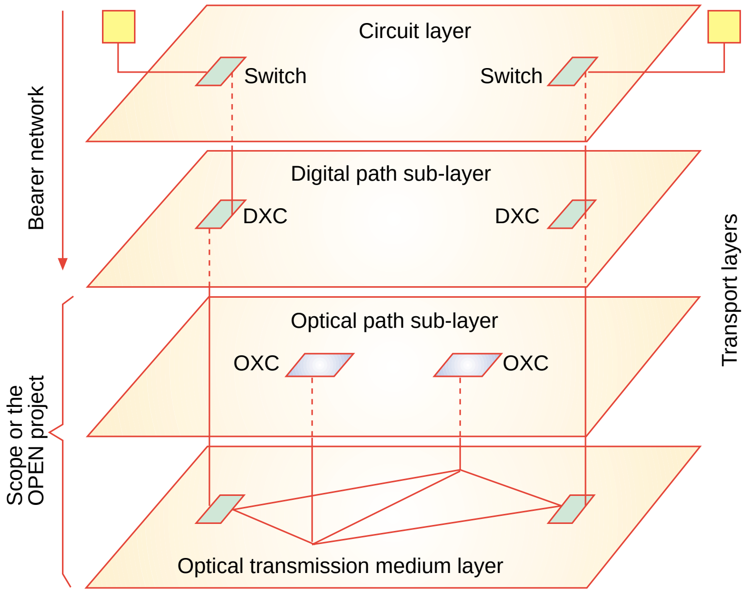


Figure 4 Open Network Concept

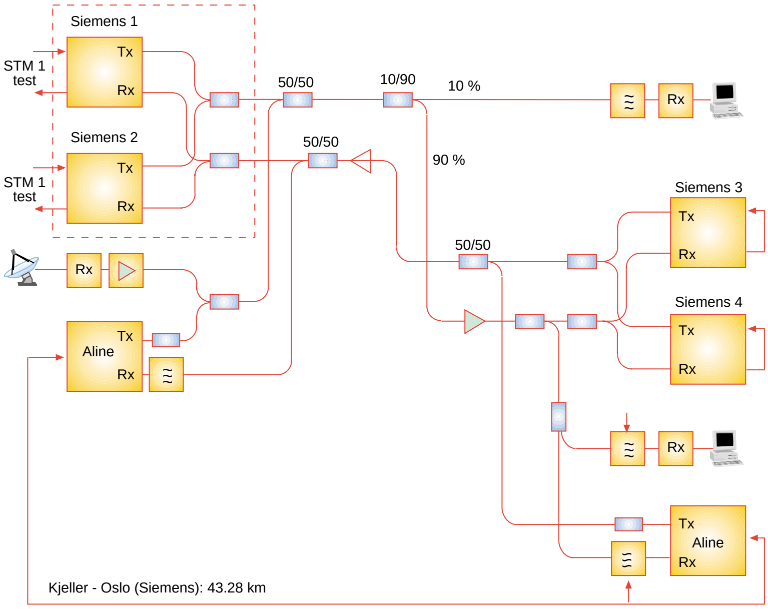


Figure 5 Bliss trial in Norway

5.2 Domain 2: Photonics technologies

The ultimate goal for a European Broadband Infrastructure is a network based upon optical fibre technology, i.e. a transparent optical network. The focus of interest in ACTS is to optimize the use of photonics by analyzing network as a whole, rather than just using optics for individual point-to-point links. Pending breakthrough is within *switching/routing* functionality, *optimal balance between optics and electronics*, multiple wavelengths as a means for *routing and multiplexing, network management*, etc.

Telenor is participating in four projects in the photonics domain, addressing several of the above mentioned issues:

- **Open:** Optical Pan-European Network
- **Bliss:** Broadband Lightwave Sources and Systems
- **Mephisto:** Management of Photonic Systems and Networks
- **Actual:** Application and Control of Widely Tunable Lasers

The last project, Actual, is a new project starting in 1998.

Open aims at studying the feasibility of an Optical Pan-European overlay Network by interconnecting major European cities by high-capacity optical fibre links. The nodes will use multi-wavelength 4x4 cross-connects with wavelength routing stages.

The potential capacity of each fibre link will be upgradable to at least 40 Gbit/s, each channel supporting STM-16 SDH or higher data-rate transport service. The proposed approach relies on extensive use of WDM for both transmission and routing purposes.

There are two field trials, one between Norway and Denmark, and one between Paris and Brussels. The Nordic field trial uses links between Arendal and Thisted/ Hjøring in Denmark with four-wave mixing and very long repeater spacing.

Mephisto considers network management (TMN) to an advanced all-optical core network. It exploits wavelength division multiplexing (WDM) for transmission and routing. The project develops a generic information model for operation and management of optical networks and their components.

Bliss has as its main objective to bring key advanced components for optical networks to full maturity. Demonstrations of practical applications are carried out within the project as well as in other ACTS projects. The project will demonstrate a 4 x 2.5 Gbit/s WDM long haul transmission system including a cross-connect where the influence of specific device properties are studied. A main focus is also on the implementation of optical receiver chips in Access Networks, where a PON and an ATM ring field trial is performed. The latter is based on the Aline concept of Siemens/Telenor.

Actual is starting up in 1998 (from the third call). It will develop a methodology for control and management of *widely tunable lasers* for WDM networks. The lasers should be capable of handling 128 channels with a 0.4 nm channel spacing. NTT in Japan is participating in the project. Their focus is on the control part of the laser.

5.3 Domain 3: High Speed Networking

The Race programmes focused a lot on broadband networking principles like ATM. At this stage ACTS wants to ensure that high-speed networks will interwork at all levels and will satisfy the user needs and demands. Projects in this domain will thus contribute to broadband systems made of different technologies and provide a platform for advanced multimedia services. The trials and experiments will allow a verification of the technologies, the user acceptance of the technologies, the Quality of Service concepts and the compliance with standards and regulations.

Telenor participates in the following 8 projects within this domain:

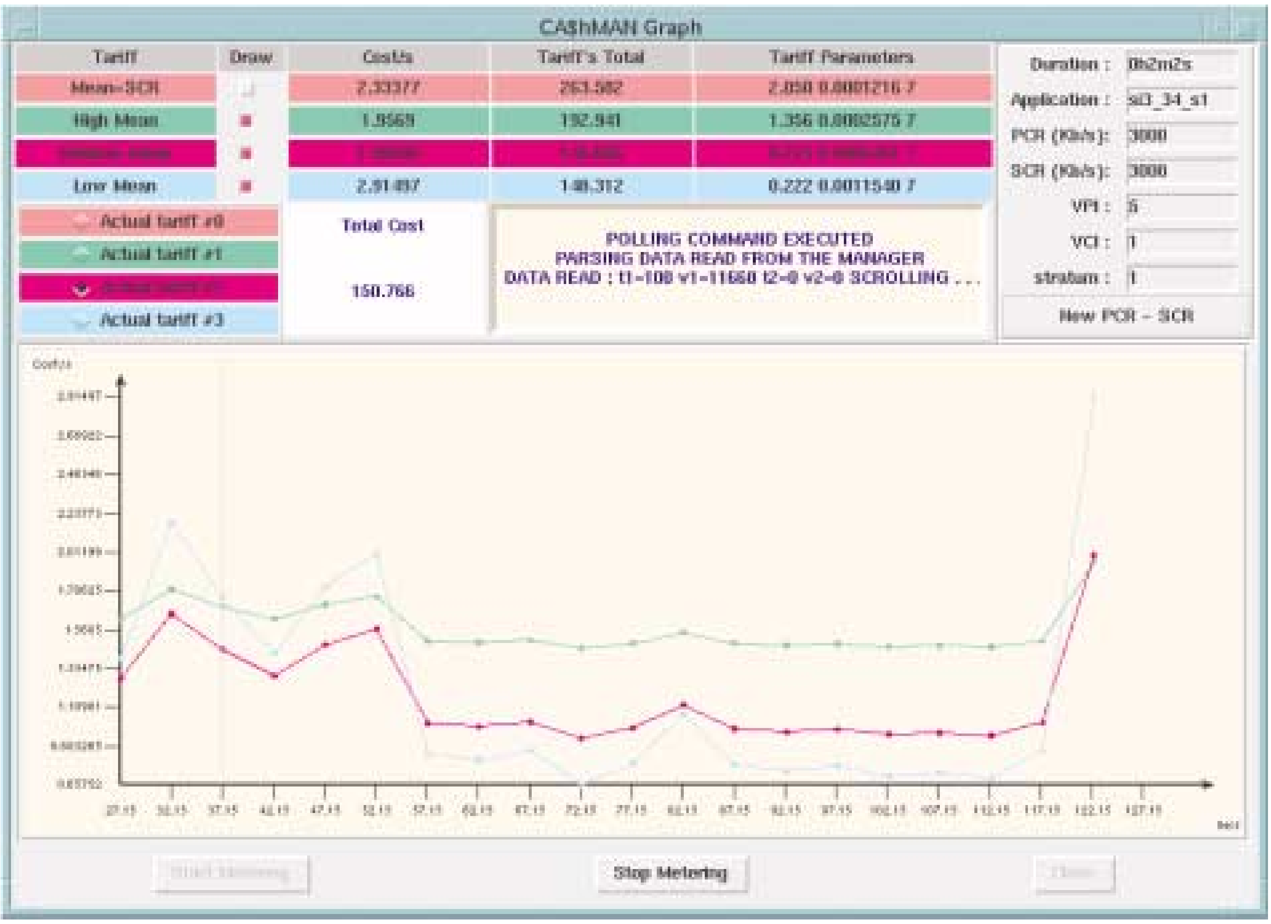
- **Asiccom:** ATM switch for Integrated Communications, Computation and Monitoring
- **Ca$hman:** Charging and Accounting Schemes in Multi-Service Networks
- **Expert:** Platform for Engineering Research and Trials
- **James:** Joint ATM Experiment on European Services
- **Nice:** National Host Interconnection Experiments
- **Reform:** Resource Failure and Restoration Management in ATM based IBCN
- **Diana:** Demonstration of IP and ATM Networking Applications
- **ITUnet:** International Trials with Users and Networks from European Testbeds.

In this listing the first four projects belong to phase 1 of ACTS and will be terminated in the first half of 1998, Reform is a phase 2 project (i.e. terminating in 1999) and Diana and ITUnet are phase 3 projects, starting in 1998.

James and *Nice* have been described in section 4 and will not be further dwelt on here.

Asiccom realizes an ATM Gigabit switch on a single-chip, developing the architecture for an effective support of a broad range of services, and implementing a Gigabit ATM testbed.

Ca$hman studies the charging and accounting schemes for ATM networks. The project develops appropriate pricing



CA$hMAN Graph

Tariff	Draw	Cost/s	Tariff's Total	Tariff Parameters
Mean-SCR	□	2.33377	263.582	2.050 0.0001216 7
High Mean	■	1.9569	192.941	1.356 0.0002575 7
[illegible]	■	[illegible]	[illegible]	[illegible]
Low Mean	■	2.91497	148.312	0.222 0.0011540 7

Duration : 0h2m2s
Application : si3_34_s1
PCR (Kb/s): 3000
SCR (Kb/s): 3000
VPI : 5
VCI : 1
stratum : 1

Total Cost
150.766

POLLING COMMAND EXECUTED
PARSING DATA READ FROM THE MANAGER
DATA READ : t1=100 v1=11660 t2=0 v2=0 SCROLLING . . .

Figure 6 Charging ATM traffic with different tariff parameters

models and their efficient implementation in hardware and software. Extensive use is made of National Host facilities for validation and for acquiring important user feedback.

Reform has an objective to specify, design and develop a system with the necessary functions for ensuring network performance and availability within acceptable levels. It particularly considers mechanisms for *fault detection, self healing mechanisms, intelligent routing mechanisms, load balancing algorithms*, to mention but a few.

Expert has as its main objective to enhance the ATM testbeds in Basel and Leichelsham. These two testbeds were established in the RACE-II programme. New features are APON access and an integrated service CPN switch. The platform is used to determine critical factors in ATM network performance and control of end-to-end QoS. The Expert platform now contains 14 ATM switches, and a variety of interworking units like ATM/ISDN and ATM/ Frame Relay interworking units as shown in Figure 7.

Diana is a starting project that specifies and provides a generic network infrastructure involving ATM and next generation IP.

ITUNet is based upon ideas coming from Telenor. It aims at experimenting with several broadband services to residential users using e.g. xDSL. It also offers an innovative network solution based on inverse multiplexing of ATM over VDSL, using twisted pair access networks.

5.4 Domain 4: Mobility and Personal Communications Networks

The projects within this domain cover three sub-areas:

- Mobile services
- Mobile/wireless network platforms
- Enabling technologies.

Telenor only participates in the second sub-area, the network platforms.

There are two different types of platforms considered in ACTS, the UMTS (Universal Mobile Telecommunications System) and the WLAN/MBS platform. Here *WLAN* stands for Wireless Local Area Network and *MBS* Mobile Broadband System.

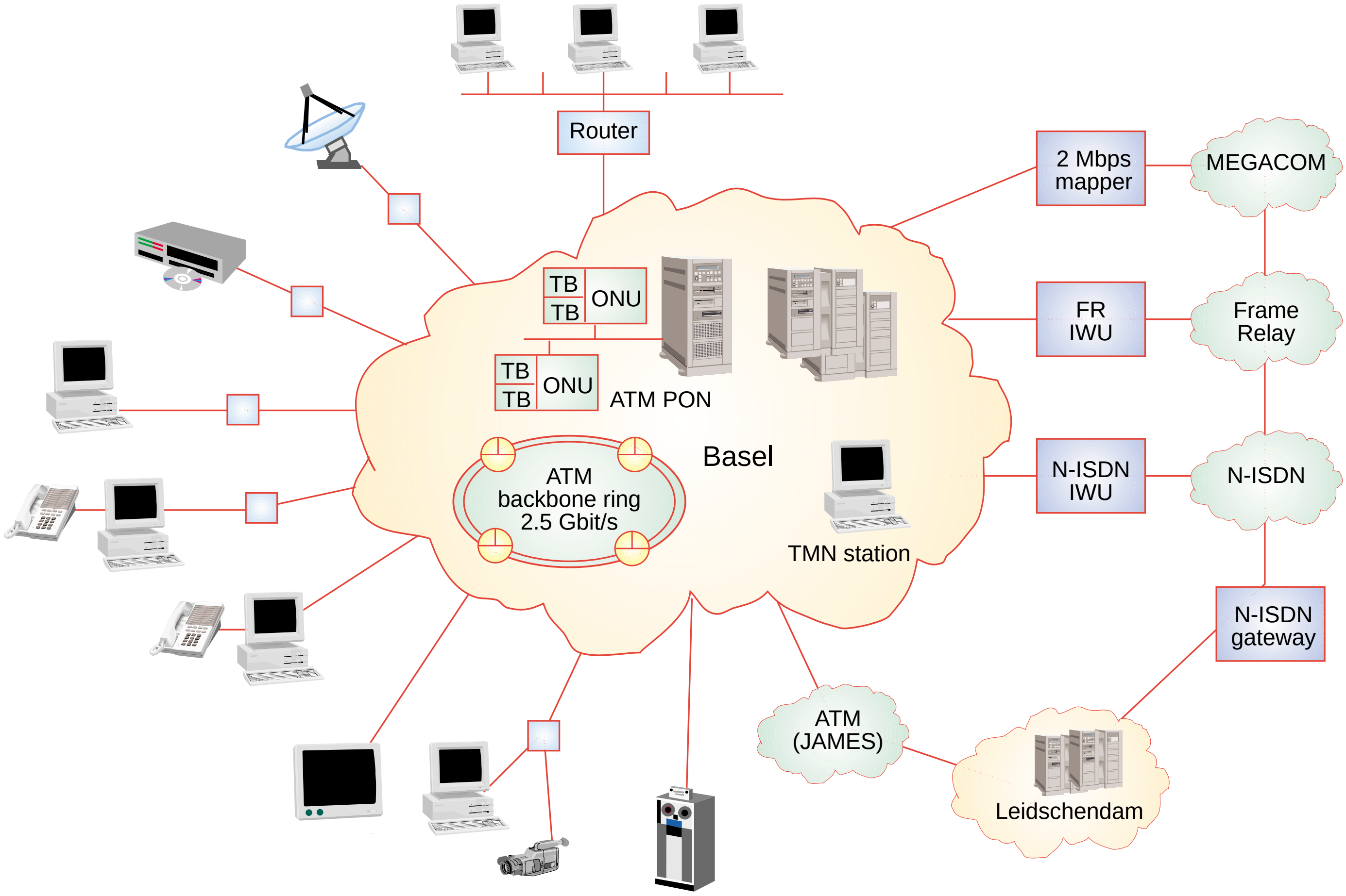


Figure 7 The Expert platform

The main characteristic of UMTS is a hierarchical cell structure designed for support of a wide range of multimedia broadband services within the various cell layers. Use is here made of advanced transmission and protocol technologies.

The driving force for introducing WLAN in an office environment is the proliferation of portable and laptop computers and the potential savings in avoiding wiring or re-wiring of buildings. Hence, next generation mobile systems must incorporate an integrated WLAN capability.

Telenor participates in the following three projects:

- **Sinus** (Satellite Integration into Networks for UMTS services)
- **Samba** (System for Advanced Mobile Broadband Applications)
- **Sumo** (Satellite UMTS Multimedia Service Trials Over Integrated Testbeds).

Needless to say, Sinus and Sumo belong to the UMTS platform and Samba to the WLAN/MBS platform.

Sinus' main objective is to validate the UMTS segment for different satellite interfaces and interworking with terrestrial components. The project also assesses the economical and technical feasibility of providing services through the UMTS satellite component. Sinus will define and demonstrate an end-to-end communication network. The trials will implement inter-segment hand-over (satellite–terrestrial), mobile management call routing, and resource management.

Sumo is a 'third call project' starting in March 1998 with a planned duration of 22 months. It is heavily based on Sinus and aims at identifying and demonstrating service support and network control for the satellite segment of UMTS. A testbed will be developed, with features like multimedia terminals, satellite access terminals, satellite channel emulator, satellite interference simulator and a live satellite channel. The Sumo trials address *Interoperability* between terrestrial UMTS networks and various satellite systems (LEO; MEO; GEO), *capacity on demand* and *validating* the GRAN concept (i.e. independence of the physical access scheme (CDMA, FDMA, TDMA).

Samba develops an MBS trial platform consisting of two base stations and two mobile stations operating in the 40 GHz frequency band. The millimetre-wave transceiver as well as the antenna are developed within the project. Each mobile station offers a transparent ATM bearer service up to 34 Mbit/s.

5.5 Domain 5: Service Engineering, Security and Comm. Management (TMN)

This domain focuses on

- how the inherent intelligence in the network may be employed to provide flexibility and open competition in the provision of services
- means for managing and administrating end-to end communication systems which span many independent network operators and service providers.

The emphasis is on applying the best techniques for system specifications and corresponding software development with the aim of creating open platforms.

Telenor participates in one project in the service engineering sub-area:

- **Retina:** An Industrial-Quality TINA Compliant Real-Time DPE.

Retina develops and demonstrates a TINA compliant Distributed Processing Environment (DPE) based on CORBA. It will demonstrate a BVPN service V1 application to be run in a wide-area distributed environment. Experiments are based on local ATM networks at each demo site, which are connected to their respective National Hosts in Norway and Italy.

5.6 Domain 6: Horizontal Actions

As mentioned before, there are in ACTS several projects that are of common interest to the Programme as a whole. Three such projects were discussed in section 4 – James, Nice and Infowin. Only the latter, however, is part of the *domain* 'Horizontal Actions' that we are now going to discuss.

The projects in this domain are grouped in the following sub-areas:

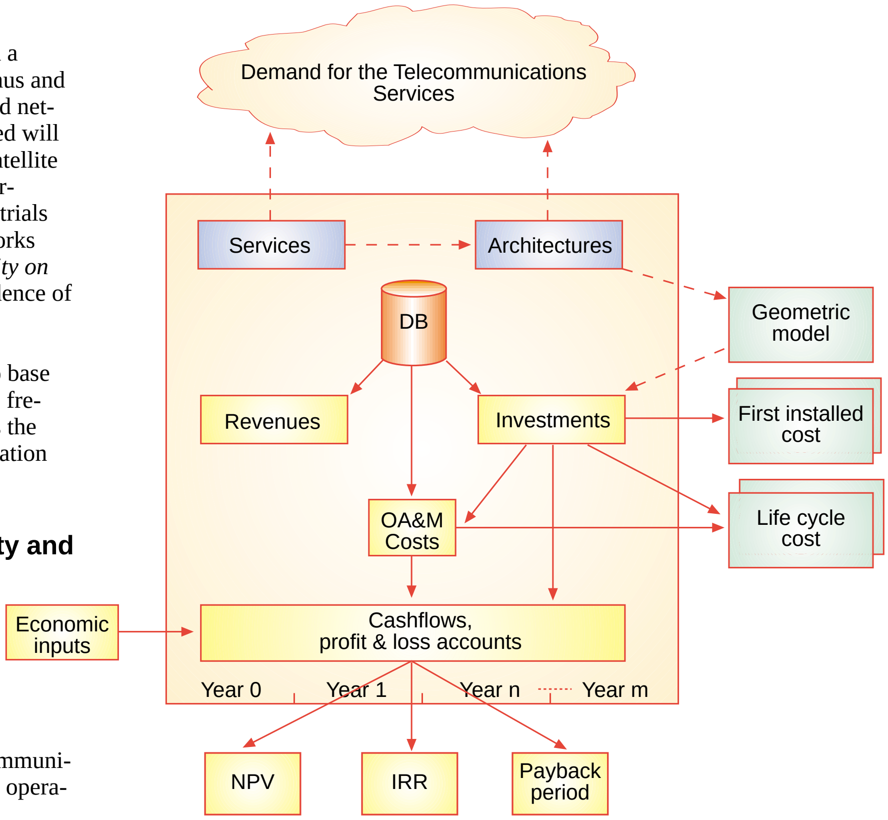


Figure 8 The Optimum platform

- Techno-Economic and Social Aspects
- Concerted Actions and Consensus Development on Guidelines
- Telework and Electronic Commerce for SMEs
- Dissemination, Exploitation and Broadening Participation.

Telenor is at present participating in the following four projects:

- **Optimum:** Optimized Network Architectures for Multimedia Services
- **Infowin:** Multimedia Information Windows for National Hosts
- **Tera:** Techno Economic Results from ACTS
- **Infobridge:** The Bridge from ACTS to the Outside World.

Here the last two projects come from the third call, starting at the beginning of 1998, and to some extent one can say that Optimum evolves into Tera and Infowin into Infobridge. Referring to the sub-division above, Optimum/Tera belong to the Techno-Economic as well as the Dissemination sub-domains. Infowin/Infobridge certainly belong to the Dissemination sub-domain.

Optimum's objective is to establish guidelines for the introduction of advanced communications network and multimedia services in a competitive environment. That is to say, Optimum considers the telecommunications access network, compares the cost (including forecast) of different technologies, like optics, copper, radio, and give guidelines for further evolution of the network. The architecture for a specific set of services is being defined and combined with traffic, demand and tariffs. Risk evaluation for various scenarios is also included in the project.

Telenor is the prime contractor of the project.

Tera is the continuation of Optimum, which will terminate in 1998. Its objective is to support consolidation of deployment guidelines and their dissemination. Tera will also perform techno-economic evaluation of outputs from various ACTS projects

Infobridge is a continuation of Infowin and intends to disseminate and promote results from the ACTS programme. This will be achieved by developing an Infospace, publication of regular news-service, publishing CD-ROM, etc.

6 Concluding remarks

Telenor has participated in altogether 24 ACTS projects during a 5 – 6 years period (plus the project Emphasis, from which we chose to withdraw after one year's participation). In the Appendix is a listing of all the 24 projects with their Telenor contact person. The experience gained through this collaboration has had an important bearing on the professional work and competence of Telenor Research and Development; the results and knowledge thus obtained have been acquired in an extremely cost-effective way. Projects in the NOK 50 – 100 mill. range are not easily financed on a national level.

It is also interesting to follow the ease with which our researchers are now taking part in the international research community. We are, together with our Nordic colleagues, highly appreciated as partners in the various EU projects and are from time to time getting more offers than we can accept. It is also clearly seen that the project consortia we enter into have been increasingly solid; in the first phase of call for papers in ACTS we had a 'hit rate' (i.e. approved projects) of about 60 – 70 %, in the last call it was very close to 100 %! Hence, the experience gained through the ACTS participation has not only been on the professional level; equally important has been the 'education' of our researcher to be part of, and feel at home in, the international research community. This is also reflected in our role in the projects; in the first phase it was rather unthinkable to go for a project leadership, in the second phase Telenor was 'prime' (i.e. project leader) in two projects (Optimum and Crabs), and in phase three we were prime in two additional projects (Tera and Itunet).

The yearly project evaluations, performed by a panel of experts on behalf of the EU Commission, show good results for most of the ACTS projects in which we participate. From the last evaluation, in January 1998, the Open project got top score on all evaluation points with MoMuSys and Expert very close behind.

Rolf B. Haugen is Chief of Research and Head of External Relations at Telenor R&D. His work includes co-ordination of Telenor activities in the EU research programme, EURESCOM and national projects with the Norwegian Research Council, as well as co-ordination of standardization activities in Telenor. He is a member of the General Assembly in ETSI and EURESCOM. In the BT-alliance he is responsible for co-ordinating technology collaborations.
e-mail: rolf.haugen@fou.telenor.no

7 Appendix

Overview of Telenor participation in ACTS projects with contact person:

ACTS Domain	ACTS Project	Telenor Contact
Interactive Digital Multimedia Services	Momusys	Robert Danielsen
	Custom TV	Robert Danielsen
	Maestro	Ola Ødegård
	Vis à Vis	Bjørn Hestnes
	Crabs	Agne Norbotten
Photonic Technologies	Open	Torodd Olsen
	Bliss	Evi Zouganeli
	Mephisto	Terje Henriksen
	Actual	Evi Zouganeli
High Speed Networking	Asiccom	Geir Millstein
	Ca$hman	Ragnar Andreassen
	Expert	Harald Pettersen
	James	Svein Tore Johnsen
	Nice	Kaare Inge Sletta
	Reform	Svein Tore Johnsen
	Diana	Egil Aarstad
	ITUNet	Einar Edvardsen
Mobility & Personal Communications	Sinus	Jørn Kårstad
	Sumo	Jørn Kårstad
	Samba	Stein Svaet
Communications Management	Retina	Eirik Dahle
Horizontal Actions	Optimum	Borgar Olsen
	Tera	Borgar Olsen
	Infowin	Thorbjørn Thorbjørnsen
	Infobridge	Thorbjørn Thorbjørnsen

InfoWin
– The Multimedia Information Window for ACTS

THORBJØRN THORBJØRNSEN AND CLAUS DESCAMPS

ACTS (Advanced Communication Technologies and Services), established under the Fourth Framework Programme (1994 – 1998), represents the European Union's major effort to support research and development in the field of Telecommunications. The work within the programme is carried out in the context of trials to encourage a dialogue between developers and users. The ACTS programme is a truly co-operative challenge that brings together many companies and organisations from all sectors of the European telecommunications industry in more than 150 projects. The success of ACTS can only be achieved through the effective exchange of information, not only within the programme, but also across the boundaries of the programme. In this context InfoWin, the ACTS Information Window, was established with the purpose of adding value to the information flow between ACTS projects and between the ACTS programme and the outside world.

1 Introduction to InfoWin

The InfoWin project provides the ACTS Information Window (http://www.infowin.org). This window allows information to flow from ACTS projects to the outside world and also allows the outside world to see what is happening in ACTS. The Information Window ensures that ACTS participants keep an up-to-date view of the development of the telecom market and its needs, whilst simultaneously ensuring the visibility of ACTS and maximising its impact. InfoWin also provides services for the internal communication of the ACTS programme.

InfoWin exists to maximise the synergies of the ACTS programme, and the economies of scope, scale and integration which can be achieved by working together on advanced communications. The increased rate of development of advanced communications services and implementations brought about by these synergies will make an important contribution to the development of the European economy. By maximising the impact of the work of ACTS, InfoWin supports the development of the information society in Europe and on a global scale. InfoWin also supports the dissemination of advanced communications concepts and services into all the regions of Europe, ensuring that the benefits can be felt throughout the community, such as in the introduction of teleworking in peripheral regions and the development of high-speed infrastructure in the core regions.

The project consortium includes National Hosts, telecommunication companies, academic institutions, computing and research centres and specialists on marketing and information dissemination.

2 How InfoWin works

InfoWin focuses on information content, and follows a scientific approach for providing the information presented at the information window. The project is structured around the key steps in the *information publication process*: information gathering and writing relevant material, editing the information in an appropriate way for different target audiences and the publication and marketing of the information to those audiences. The electronic information system activities included in the project are those necessary to support the information dissemination.

InfoWin aims its products at different target groups. These range from the inner circle of the ACTS community over the general R&D, business and political communities, to the general public at large. The work is based on an in-depth understanding of the different communities that have an interest in the ACTS programme and projects. Major contributors to this work are National Hosts and educational institutions.

The regular products of InfoWin are: Newsclips, Bulletins, Thematic Issues and the ACTS yearbooks (all described in Chapter 3). The effort that goes into the production and dissemination of these publications is structured as:

- Information gathering and reporting (based on origin)
- Editorial work (based on subject)
- Dissemination – marketing and promotion (based on target audience).

The next sections will deal with these points in greater detail, starting off with a description of the editorial work of InfoWin.

2.1 Editorial work

The purpose of editorial work is to add value and structure to the raw information. Value is added by analysing and consolidating the information so as to present the information in forms that are easily accessible and in line with the needs of the intended target groups.

Much effort goes into tailoring the publications for the different target groups such as:

- *The ACTS community.* This community is well aware of the programme, has good knowledge of the main content of the programme and requires technical and detailed information regarding ACTS and research activities elsewhere.
- *The R&D community in general,* such as research institutions and universities. This community is aware of the existence of the European research programmes in various levels of detail. A group that requires detailed, technical information.
- *The business community:* industry, financial services, administrations and users. They have to keep up-to-date with new developments in the advanced communications area.
- *The political community:* administrations and technical bodies in charge of implementing policies. The quest for information is crucial in their forecasting and regulatory roles.
- *The public at large.* Proper information can improve the general understanding of upcoming new technologies and services.

The information enhancement pyramid shown in Figure 1 applies to the publications of InfoWin, and it depicts the amount of editorial work that is put into the different publications. The material closer to the top of the pyramid represents the publications that require the highest editorial effort. The bottom of the pyramid represents material basically dealing with raw information of the form of project deliverables and reports. On the left we find the different kinds of publications represented by the pyramid.

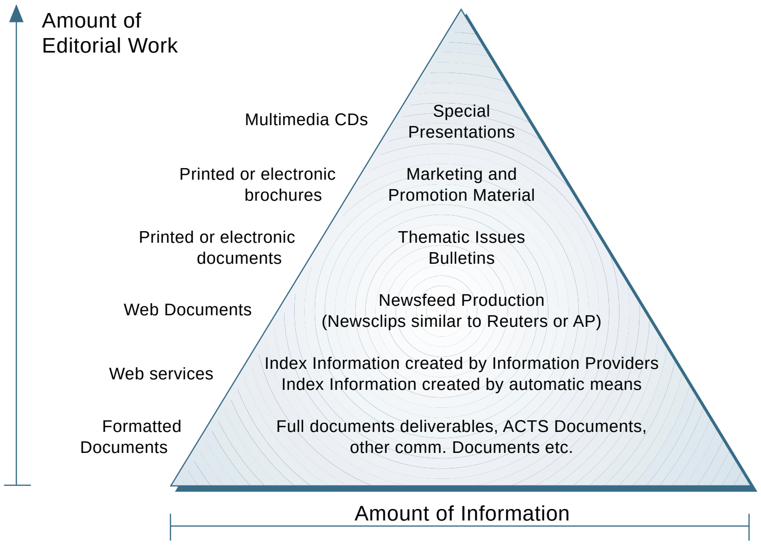

Amount of Editorial Work
Multimedia CDs
Special Presentations
Printed or electronic brochures
Marketing and Promotion Material
Printed or electronic documents
Thematic Issues Bulletins
Web Documents
Newsfeed Production (Newsclips similar to Reuters or AP)
Web services
Index Information created by Information Providers
Index Information created by automatic means
Formatted Documents
Full documents deliverables, ACTS Documents, other comm. Documents etc.
Amount of Information

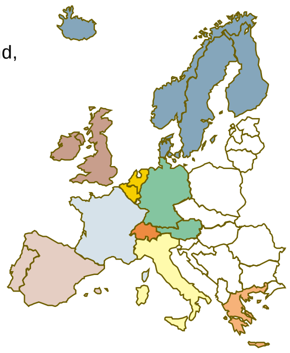
Figure 1 Information enhancement pyramid

2.2 The regional representatives – a key mechanism to InfoWin

To support the vast information gathering and dissemination activities, InfoWin established the regional representative mechanism.

InfoWin aims at bringing services close to the people who need it. To support the vast information gathering and dissemination activities, InfoWin established the regional representative mechanism. Western Europe has been divided into nine regions, each covered by one or more regional representatives responsible for promoting the activities of ACTS and for gathering information on relevant results and activities.

- UK and Ireland
- Nordic countries (Denmark, Finland, Iceland, Norway and Sweden)
- Germany and Austria
- Greece
- Italy
- Portugal and Spain
- Switzerland
- Benelux.
- France



InfoWin's Regional Representatives are a group of regional experts that provide an important interface between the ACTS community and the outside world. These 'local correspondents' perform a dual role in gathering and producing information, and marketing ACTS/InfoWin and disseminating information.

Through personal visits, attendance of workshops, conferences and other events, regional representatives establish and maintain contacts within the ACTS R&D and business communities.

2.2.1 Information gathering by regional representatives

The objectives of the information gathering activity is to:

- collect information and handle requests for information collection
- gather information that directly relates to the ACTS programme and make this information available for further processing
- working as technology journalists, cover conferences and meetings related to ACTS projects and the programme as a whole
- report on ACTS projects, their progress and results.

InfoWin aims to improve the flow of information between ACTS projects, and between ACTS projects and the outside world. The first task of regional representatives is to gather the information within the boundaries set by the editorial board of InfoWin. The regional representatives have therefore to keep abreast with the developments in advanced telecommunication technologies and to understand which relevant research is going on within their region and in ACTS projects.

2.2.2 Dissemination and marketing activities by regional representatives

The main goal of dissemination and marketing is to deliver the right information to the right audience. Through their close contact with the local market, the regional representatives provide a good mechanism to achieve effective dissemination. They also receive valuable feedback or requests for specific information through their contact network.

The target groups of InfoWin have very different requirements. Through the regional representatives, InfoWin carries out large-scale surveys to identify their needs. The surveys are usually carried out at concertation meetings. Concertation can be defined as the bringing together of people and their projects, to profit from each other's knowledge and experience, to co-ordinate or encourage the convergence of on-going work on the most important issues. The aim is to build a broadly-based consensus on the ways to realise advanced communications in Europe. Concertation takes place both within ACTS and externally with related initiatives and interest groups representing the broader communications public.

In ACTS, the concertation mechanisms are organised around Domains and Chains. Domains provide internal forums for projects working on similar technical areas to share ideas and experiences. Chains are objective driven and more outward facing than the Domains, and comprise projects working together on technology development and technology trials.

2.2.3 The Regional Representative Community

In order to make the best use of the experience and knowledge that exist in the Regional Representatives network, a free-flowing information exchange and personal rapport with fellow regional representatives is essential. This open infrastructure of contact points allows for rapid exchange of information, and makes for a global overview of the ACTS programme.

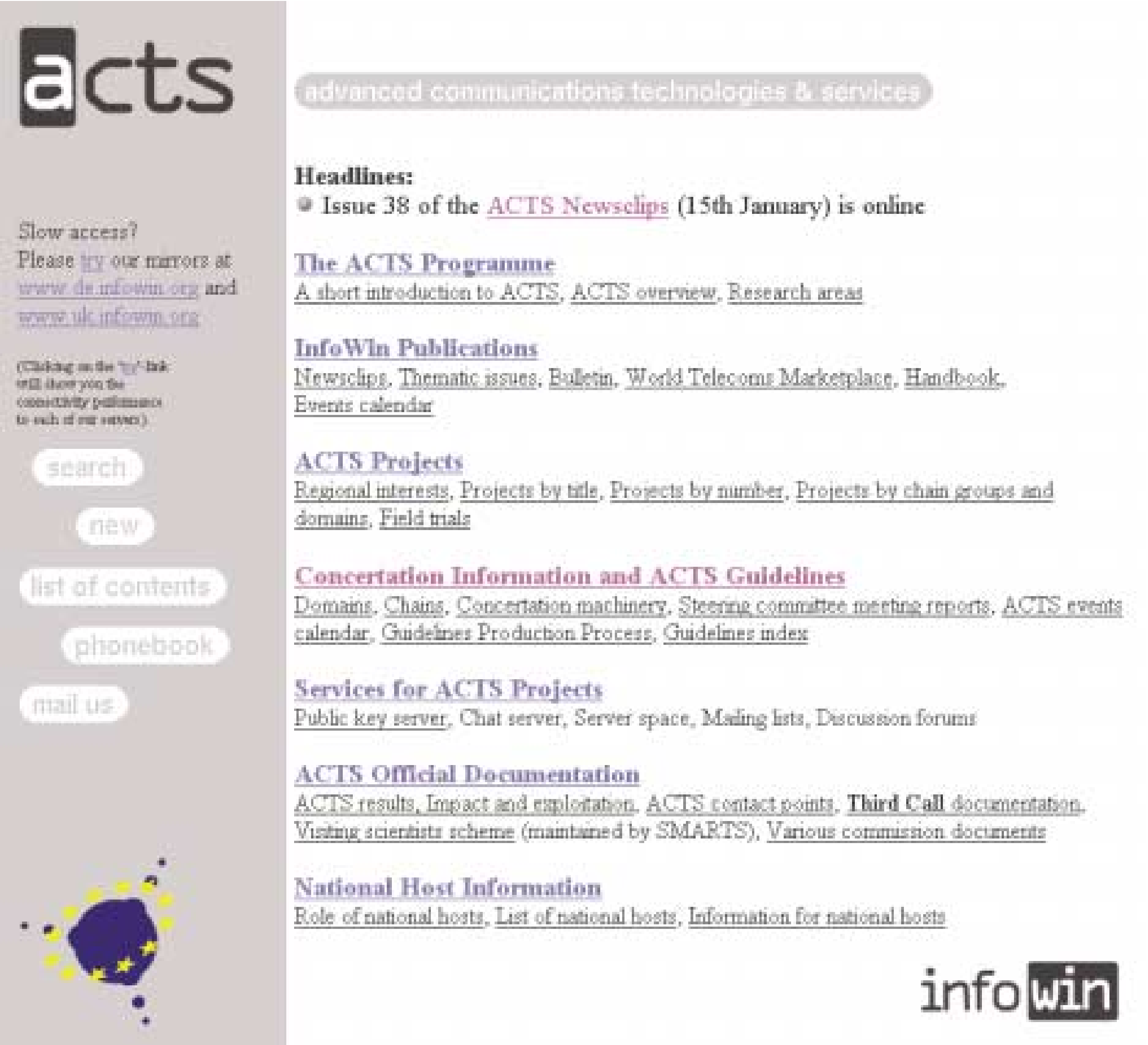
Within each region, the regional representatives operate as teams. Each Region has a main contact person who co-ordinates and monitors the work in her/his region. The co-ordinator assigns work to members of the team and is the main contact point within the region for enquiries from both inside and outside ACTS.

3 Products

Most of the InfoWin products are available through the ACTS Information Window. InfoWin also produces booklets, manuals and CD-ROMs, which are targeted at different audiences and at manifold events.

3.1 Information Window

The Information Window is a repository of all general information and public domain results of the ACTS Programme. It resides on the WWW and makes full use of a decentralised architecture to allow the Commission, each project, Domain and Chain groups to manage their own contributions. InfoWin ensures the overall coherence of the information window, and maintains a single, well-structured point of entry. The information window is a basic resource that supports ACTS concertation, both internally and externally to the programme. It simultaneously supports the development of working documents in closed discussion forums, and broadens awareness of ACTS results through regular newsletters and dissemination initiatives on specific issues.



Slow access?
Please try our mirrors at www.de.infowin.org and www.uk.infowin.org

(Clicking on the 'try'-link will show you the connectivity performance to each of our servers.)

Headlines:
- Issue 38 of the ACTS Newsclips (15th January) is online

The ACTS Programme
A short introduction to ACTS, ACTS overview, Research areas

InfoWin Publications
Newsclips, Thematic issues, Bulletin, World Telecoms Marketplace, Handbook, Events calendar

ACTS Projects
Regional interests, Projects by title, Projects by number, Projects by chain groups and domains, Field trials

Concertation Information and ACTS Guidelines
Domains, Chains, Concertation machinery, Steering committee meeting reports, ACTS events calendar, Guidelines Production Process, Guidelines index

Services for ACTS Projects
Public key server, Chat server, Server space, Mailing lists, Discussion forums

ACTS Official Documentation
ACTS results, Impact and exploitation, ACTS contact points, **Third Call** documentation, Visiting scientists scheme (maintained by SMARTS), Various commission documents

National Host Information
Role of national hosts, List of national hosts, Information for national hosts

Figure 2 Homepage of the InfoWin Infospace

3.2 On-line Information

- InfoWin Infospace (www.infowin.org mirrored at www.de.infowin.org)

  The Infospace is the main entrance to information about ACTS-projects, Chains, Domains, ACTS in general, National Hosts, etc. It contains information about the latest news and reviews regarding public domain results and working papers. A screen-dump of the homepage of the ACTS Infospace is shown in Figure 2.

- ACTS Newsclips

  The ACTS Newsclips activity produces a bi-weekly column with spot news, covering developments in Advanced Communications Technologies and Services. Newsclips are published on-line within the Infospace and also via e-mail.

- Thematic Issues

  InfoWin Thematic Issues provide in-depth coverage of special research topics within ACTS and other topics of current interest within the area of trans-European telecommunication networks and the information society. Topics are often user-oriented or service-oriented. They are published in various formats (paper, CD-ROM, on-line). To date, the following Thematic Issues have been published:

  - *Multimedia Success Stories:* A series of case studies of multimedia applications that are commercially available. This thematic issue provides an insight into the general factors that contribute to the success of a multimedia project. Whatever your interest in multimedia might be, this product is meant not only to inspire you, but also to show you how a vision can be turned into commercial reality.
  - *Advanced Business Communications in Europe:* The implementation of new technologies is about to radically alter the way companies communicate, both internally and with the outside world, with customers, subcontractors and suppliers. This thematic issue comprises case studies of European Small and Medium sized Enterprises (SMEs) that make extensive use of advanced business communication technologies and from whom other SMEs can learn relevant lessons.
  - *Multimedia Broadcast:* Provides an overview of European R&D in the field of interactive multimedia broadcast services and applications.
  - *Mobile Communications On-line:* This publication exploits the pervasiveness of the WWW and the Internet to increase the awareness of European R&D and standardisation activities in the expanding field of Mobile Communications. Apart from evolution issues, where experts survey the main research topics, this product provides state-of-the-art information concerning European mobile operators and services.
  - *ATM in Europe:* Gives an overview of the European R&D in the ATM (Asynchronous Transfer Mode) area. It focuses on the activities, trials, and results of the ACTS Programme, the TEN-IBC Preparatory Action, and other EC-funded programmes.
  - *Information Brokerage:* Gives an overview of European R&D in the area of Information Brokerage in the context of Electronic Commerce, focusing on the activities, trials and results of the ACTS Programme.

- Communication Handbook (in co-operation with the ACTS project NICE)

  The Communication Handbook is a user guide with precise and pragmatic recommendations for the use of formats and rendering tools. The services provided by InfoWin and NICE are described together with the best way to access them. These services are not restricted to ACTS participants and everyone is encouraged to try out and test them.

- Bulletin

  The InfoWin Bulletin is published quarterly, and is arranged to coincide with the ACTS Concertation meetings in Brussels. It is the in-house journal of the ACTS community and is available in various formats: WWW, paper, CD-ROM and floppy disk. It covers special research topics of ACTS and other topics of current interest, important events, articles, and publications

- ACTS yearbook – the ACTSxx Products

  Each year InfoWin assists in the production of the ACTS yearbook (published in CD-ROM and printed versions). The scope of this publication is not just to present the yearly progress of the ACTS programme, but also to give an overview of the whole programme, including a description of the

List of participants in the InfoWin consortium

Country	Organisation	Website
D	RUS, University of Stuttgart	http://www.rus.uni-stuttgart.de/
UK	Analysys Ltd.	http://www.analysys.com
F	Expertel	http://www.expertel.fr
I	CSELT SPA	http://www.cselt.stet.it
D	Deutsche Telekom Berkom GmbH	http://b5www.deteberkom.de/
D	Fraunhofer Institute for Computer Graphics	http://www.igd.fhg.de
F	INRIA	http://www.inria.fr
A	Techno-Z FH F&E	http://www.newmedia.at/main.html
GR	Intracom Hellenic Telecommunications Ele	
GR	NCSR Demokritos	
IS	PTT Iceland	http://www.simi.is
CH	Swisscom	http://www.swisscom.com
B	Synergetics IT consultants NV	http://www.synergetics.be
E	Telefonica Investigation y Desarollo	http://www.tid.es
N	Telenor FoU	http://www.fou.telenor.no
UK	Sheffield Hallam University	http://www.shu.ac.uk
DK	UNI-C, Danish Computing Centre for Research and Education	http://www.uni-c.dk

political and social background. The publication is intended for all interested parties of ACTS, including ACTS programme participants, researchers, politicians and the public at large which is involved in ACTS through user trials. ACTS 97 is the annual technical report of the EU's research and technological development programme on Advanced Communications Technologies and Services. The report is published in seven volumes:

- *Brochure:* a brief introduction to the ACTS Programme
- *Compendium of Practical Experimentation and Trials:* an overview of the experimentation undertaken by ACTS projects. The technical platforms are specified and details on where and when the trials are taking place are given.
- *Overview:* This is an executive summary of ACTS implementation, and the broader context for its results
- *Programme Guide:* Programme level results, and ongoing concertation activities within the Domains, Chains and Chain groups
- *Project Summaries:* a summary of each ACTS project
- *fACTS Booklet:* a handy, pocket-sized booklet containing essential reference information for programme participants
- *ACTS 97 CD-ROM:* the complete ACTS 97 publications in multimedia form with additional information on a number of related subjects.

4 Achievements

The InfoWin projects and services are in great demand, witness the high number of monthly hits – on average 200,000 – on the InfoWin servers. There are over a thousand subscribers to the InfoWin Newsclips, which further demonstrates the value of the service.

InfoWin has received two awards for its work.

4.1 European Telework Award

European Telework Week, initiated by the European Commission in 1995, is an annual event that supports industries, administrations and other interested parties in the teleworking arena.

Telework Awards are given to organisations that, in the opinion of the evaluators, have made outstanding contributions to telework development in Europe. The Telework Awards are supported by the partners in the European Telework Week initiative which include the European Commission, the European Telework Development (ETD) project and industry representatives.

There were over sixty nominations for the European Telework Awards '97. InfoWin received the Telework Award in the category 'media coverage' for its overall coverage of the 'European Telework Week'.

4.2 NBNSOFT

NBNSOFT, an e-journal that awards for content and stories, awarded the InfoWin Newsclips – The best "what's newest on the Net". The award was received in July 1996 in the category of the 'newest technology and educational resources'. The accompanying statement was: "ACTS News-Clips, an informative bi-weekly newsfeed about developments in Advanced Communications Technologies and Services (ACTS), cover stories ranging from satellite links in Greece and Norway to the future of PC-TV".

Thorbjørn Thorbjørnsen is Senior Adviser in the Unit of Satellite & Radio Systems at Telenor R&D, Kjeller.

e-mail:
thorbjorn.thorbjornsen@fou.telenor.no

Claus Descamps is Research Scientist in the Unit of Satellite & Radio Systems at Telenor R&D, Kjeller.

e-mail:
claus.descamps@fou.telenor.no

FERT
– Forum for European R&D in Telecommunications

TOR M. JANSEN

Introduction

In telecommunications, the service providers, the network operators and the telecommunications industry dominate the commercial and technological scene in Europe. The European Commission is helping to establish research and development programmes in many areas, where the activities are partly funded by the European Union (EU). Many of these projects involve, or are directly related to telecommunications. In many areas, the different actors in the telecommunications sector have common interests. Therefore, there is a need for making these interests influence the contents and direction of the R&D programmes.

This has resulted in the creation of several co-operative fora. The telecom operators co-operate in ETNO, European Telecommunications Network Operators. The industry has an organisation called ETIC, European Telecommunications Industrial Consortium.

FERT is an organisation with membership open to different sectors like manufacturers, network operators, service providers and users. It is the only organisation in Europe where consensus on strategic guidance to the development of telecommunications in Europe is being achieved between the different actors groupings.

The results and contributions are based on the wide range of knowledge and experience of the members, and FERT provides important input to the EU Commission and other bodies.

The actors use the forum as a platform for promoting and influencing directly and indirectly the content and direction of the ongoing and future European Union collaborative R&D programmes and contributing to the other European bodies.

Activities

The main activities of FERT is to establish

- Ways and means by which collaborative R&D in telecommunications will benefit the evolution of Europe's information society
- "Common visions" of the evolution of telecommunications based on customer requirements and advances in technology.

FERT members have initiated strategic projects in the area of telecommunications (CONCORD 2, CONVAIR) for the ongoing 4th Framework Programme of EU. The common main goal of the two projects is to help pave the way for the Information Society of the future.

CONCORDIA (CONCORD 2) – within the Telematics Application Programme, has the main task of identifying users' requirements necessary from future communication services and networks indicated by the emerging applications.

The main objectives of CONCORDIA include:

- Recording of harmonised results in Infrastructure Evolution Recommendations (IER)
- Recommendations
- Promotion of the exploitation and use of results
- Collection of feed-back on the results and of the selected solutions
- Providing advice and recommendations based also on the results achieved by the CONVAIR project in ACTS
- Supporting Sector Actors to orient their strategies and to optimise their support to the Telematics Applications.

CONVAIR – within the ACTS (**A**dvanced **C**ommunications **T**echnologies & **S**ervices) programme, has the main task of developing "Harmonised visions" of EU communications perspectives and to identify the key areas that are critical for the realisation of the future Communication and Information Society. In particular, CONVAIR aims at preparing outputs which may be used as planning guidelines for players in ICT (Information and Communication Technology), developing common statements on strategic European telecommunications evolution. Topics and priorities for future collaborative R&D Programmes will also be identified. The project's initial results are already emerging. In particular, the approach and a framework for the whole study have been defined in detail, a sound basis for discussion on visions has been set up and the critical areas and key issues have been listed during the first few months of activity.

The members of FERT are engaged in many projects within the EU R&D programmes ACTS, TELEMATICS and ESPRIT, which are within the 4th Framework Programme.

At present, a main activity of FERT is to establish "Common Visions" for the 5th Framework Programme.

Structure of FERT

Plenary Assembly

The Plenary Assembly is open to all members of FERT, and here the members may exchange information and ideas about telecommunications development and trends.

Generation and elaboration of guidelines, perspectives and proposals for the Management Board are also important tasks.

FERT Management Board (FMB)

This is the executive body of FERT. It is responsible for the establishment of Working Groups and projects, and also actions towards the EU Commission and the public. Presently, it consists of ten members from the industry and the network operators. The chairmanship is rotated every six months between the

Member Groupings represented. In the second half of 1998, Mr. Cordaro of Telecom Italia is Chairman.

Member Grouping

The members of FERT organise themselves into Member Groups consisting of the actors in the different Sectors of Telecommunications. The Sector of Telecommunications Manufacturers is already organised in ETIC. The members in the Sector of Telecommunications Operators come mainly from the ETNO organisation. The Member Groups decide about their representatives within the FERT Management Board.

Membership of FERT is open to:

- European Telecommunication Manufacturers
- European Network Operators
- European Telecommunication Service Providers
- European Users Associations.

FERT as an open forum is prepared to accept membership of all European actors in telecommunications.

Present Members of FERT Management Board

The Board at present consists of representatives from the following organisations:

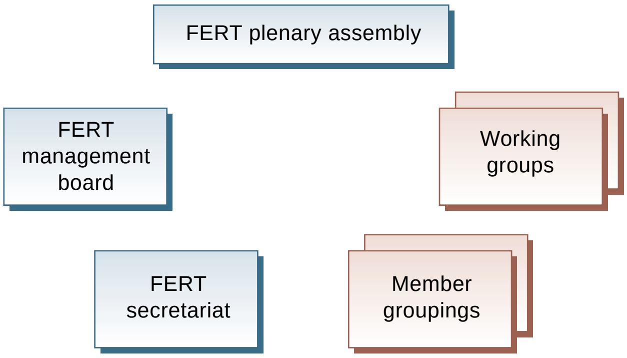
Manufacturers (ETIC):		**Network Operators:**	
GPT	(D.J. Cleobury)	France Telecom	(D. Thebault)
Alcatel	(P. Goossens)	Telecom Italia	(G. Cordaro)
Italtel	(G. Fabri)	Deutsche Telekom	(F. Wichards)
Siemens	(K.U. Stein)	BT	(J. Grierson)
Nokia	(K. Baughan)	Telenor	(T. Jansen)

FERT Secretariat

The Deputy Director, Mr P. Polese, heads the Secretariat and the address of the FERT Secretariat is:

Rue Montoyer 39, bte 6/7, B - 1000 Bruxelles
Tel.: + 32 2 549 08 40
Fax: + 32 2 549 08 52/53
E-mail: fert-info@lists.etic.be
Web site: http://www.cselt.it/webhost/fert/

For more information, either contact the Secretariat or visit the web site.



Structure of FERT

Tor M. Jansen is Director of Research at Telenor R&D, Kjeller, with the External Relations (ER) group, and also with the Mobile Communications group. He is responsible for co-ordinating the work in ETNO R&D WG, EURESCOM and FERT.

e-mail:
tor.jansen@fou.telenor.no