INTEGRATED SERVICES ENGINEERING

P L Reynolds and P W Sanders

Network Research Group, University of Plymouth, UK

1: Introduction

Integrated Services Engineering (ISE) is a relatively new branch of telecommunication engineering and is associated with the cost effective creation, deployment and management of services in a heterogeneous network environment. The conceptual separation of bearer networks and user services is now well established and forms the basis of the Intelligent Network Architecture currently being implemented by European and US Telco's. This separation is being driven, not only by the technological need to manage complexity, but also by the deregulation and open network legislation, which is leading to a multiplicity of service and bearer network providers. In such an environment, service engineering requires a process cognisant to this changing market place.

Within the EU-RACE programme, Project Line 5 addresses issues arising from the particular needs of service engineering. The basis of their work is an architectural framework that utilises the Open Distributed Processing-Reference Model (ODP-RM) [1] and encompasses the concepts of Telecommunications Management Networks (TMN) [2] and Intelligent Networks (IN) [3]. The progression of ISE started with the definition of an ODP-RM based ISE framework in ROSA (Race Open System Architecture) [4], continued with the definition of an architecture, in Cassiopeia OSA(sys) and OSA(app) (Open System Architecture-System and Open System Architecture-Application) [5] and continues to be developed with a distributed processing environment based upon OSA(sys), called the ISE Engineering Platform [6].

2: The ISE Reference Model

A reference model (RM) is an abstraction for all possible system solutions in a given domain. The RM defines the solution space or boundaries within which it provides the terms and concepts for exploring the specific problem. The RM provides the basis of the language used to both communicate and understand the concept by providing a common background, a set of values, etc. The RM comprises an outline framework, a set of models and makes reference to any standards within the domain.

A framework is provided so than an appropriately structured architecture can be developed that is logically organised in a way that is relevant to the domain being addressed. The models allow discussion and reference to be made for information, opinion and/or decision making. The standards allow for imitation, form, style or pattern.

The reference model for ISE is the ODP-RM. It provides a framework which enables designers to construct distributed applications without having to take into account the diversity of hardware. The ODP-RM Engineering Viewpoint focuses on the infrastructure and functions required to support distributed processing and addresses modularity, transparency, management, autonomy and scaling. The difference between the Engineering Viewpoint and the Computational Viewpoint of ODP, and hence ISE, can be characterised as being the distinction between 'what needs to be done' (computational viewpoint) and 'how it is to be achieved' (engineering viewpoint). The model provides designers of operating systems and communication experts with a description of an ODP-RM in terms of the control and transparency mechanisms, and communications networks that enable the service distribution. The communication functions are visible in the engineering viewpoint, so an engineering language is defined to enable the specification of the processing, storage and communication functions required for implementations, that are based upon ODP-RM.

The syntax and semantics of the engineering language are object based and include:

- binder objects - they have three interfaces, an interface to the stub (below), an interface for interaction with a protocol object, and a control interface;
- capsule - is a set of objects forming a single unit for processing and storage, e.g. a virtual machine. It

corresponds with the notion of address space or process in computing systems. A capsule is always situated at a single node;

- cluster - a set of objects which form a single unit, e.g. memory;
- communication objects - they interact with interceptor objects when the communication takes place between domains, e.g. an inter-sub-network relay;
- engineering objects - they need the support of a distributed structure and interact with each other by way of their local transparency and nucleus objects;
- interceptor objects - they correspond to the notion of gateway, agent or monitor objects;
- node, a set of objects forming a single unit in space, e.g. a switch. A node comprises one nucleus and a set of capsules;
- nucleus - an object which co-ordinates functions for use by other objects;
- protocol objects - they provide communication functions. Each has two interfaces, one to interact with a binder and a communication interface;
- stub objects - they are objects which provide conversion functionality. Engineering objects are bound to stubs. A stub has three interfaces, a presentation interface to the object it supports, a control interface for quality of service management and an interface for interaction with a binder object.
- transparency - requires supporting services, for example, if engineering objects move location a means of both recording and discovering current location is required. Supporting services are modelled as objects.

In general, an engineering specification:-

- defines the roles of the different objects,
- describes the organisation of an abstract infrastructure for enabling the execution of the ODP,
- identifies reference points,
- identifies the abstractions required to manage physical distribution.

The ODP-RM defines objects in terms of abstraction and encapsulation - as 'black boxes', and a set of observable actions at their boundaries. It is the interactions between objects and their individual characteristics or observable actions, which specify the behaviour, irrespective of the objects' internal structure. Objects can be used to divide the problem into a number of distinguishable elements and allow each to be treated independently. The same object may be implemented in a number of ways in different environments, but each implementation can support the same service. The abstraction is achieved by focusing upon the behaviour of the objects. All objects interact in the same way; encapsulation ensures that the information contained by an object is accessible only through invocation of the service offered by the object. Interactions between objects and their environment are ruled by contracts. A contract constrains the co-operation between a set of objects. In the sense a contract is a dynamic specification of the configuration of the object. As an example, a particular contract deals with the Quality of Service (QoS) attributes of an interaction that are related to the invocation, such as timing constraints, resolution, load, reliability, rate of information transfer, the latency, the probability of a communication being disrupted, the probability of system failure, the probability of storage failure, etc. QoS rules may be needed: to identify relevant quality attributes of resources, to express qualitative measures for the identified attributes, to express subjective ordering of perceived user satisfaction, to express QoS negotiation, etc.

In a system involving ODP-RM, objects are responsible for their own management. They may be requested by a management application, e.g. TMN, to modify their own behaviour. An object must therefore provide a management function which permits its normal activity to be monitored and possibly modified. Deciding what is management and what is the normal function of an object may be rather difficult. One criterion is that management of the object should not be critical to its normal operation. For example, a management function could include monitoring of its performance, and changing the number of buffers it uses. Similarly use of encryption is considered part of a security service, but distribution of the keys used for encryption would be a management function.

Objects are communicated with, and used by way of, their interfaces. An interface represents a set of operations. Operations are grouped into the same interface when they are likely to be used by the same client. For example, the management operations and non-management operations would normally be separated into different interfaces. Objects may possess a number of interfaces. Interfaces are either created as part of the process of instantiation of a given object or they are created as additions to some existing object. An interface and its object work by a process called hiding. For example, in a directory object, the request, reply and error actions in the definition of the directory object are hidden at its management interface. That is, all non-management actions are replaced by an internal action called an i-action. At the functional interface the update, change security actions, etc. would be hidden and replace these with i-actions. In summary interfaces can be derived from objects by hiding all irrelevant actions.

Objects are the primary building block, not interfaces. A distributed application can be regarded as a configuration of objects, some of which are composite. The design of such an application involves identifying the objects and configurations, and, if they do not exist, designing them.

The design can be split into several stages, and involve:

- object composition:- by way of a configuration specification which provides a description of the software structure of a system, i.e. a composite object, described in terms of its constitute objects and their interconnections;
- object decomposition:- knowing how to decompose an object is not easy. There may be a number of decompositions which are equivalent to the original object. As an example, take the development of a communication object. Initially, two objects are modelled as interacting directly. The two objects can themselves by decomposed into two objects, each with one object providing the communication function. The two derived communication objects can be combined into a single communication object that hides the original interface between the two original objects.

There are a number of extensions necessary to the ODP-RM if it is to be used for ISE. These include:

- the development of the managed object concept in which objects are expanded so as to manage the resource aspects of a distributed service, which covers:
  - accounting management (effective deployment so as to maximise return on investment);
  - resource management (efficient deployment of resources so as to minimise cost);
  - service engineering management (management of risk associated with maintaining a service level agreement);
- the definition of a building block, which is usually decomposed into functional elements, network elements and chargeable units.
- the separation of service provision into two separate phases. In the first of these, the service is offered to the user and the result is a selection of the service (selection). The second phase is the establishment and use of an instance of the service which includes both service delivery and the generation of billing information (invocation).

3: The ISE Architecture

An architecture is the materialisation of style. An architecure is not a design for a domain, in the same way that gothic architecture is not the same as the Houses of Parliament, but provides a qualitative definition of style, by giving the base specification, functions and guidelines for the decomposition of functionality into style components. These components provide the wherewithal necessary to generate a particular solution that will reflect this style. Four components are provided by the architecture:

- building blocks and tools,
- guidelines, which help the designer reach answers or give answers in an unknown situation.
- recipes, which give advice on how to build subsystems with certain properties, and
- rules, which constrain the building block combinations.

Thus the architecture provides the art and science of designing with components that will produce an observable style, down to a granularity that is not cognisant of the internal realisation. This allows the designer to focus on what a system is meant to provide rather than how it is internally organised. Hence the architect has no interest in the technology that provides the building blocks! In linguistic terms the architecture provides the language syntax, i.e. the structure and form, grammar and rules governing the properly formed components of a logically based language.

Two parallel architectures exist for considerations. The first is the RACE/OSA and the second TINA-C (Telecommunication Information Networking Architecture Consortium) [7]. The latter is a consortium formed by network operators, telecommunication equipment and computer suppliers. Both are based upon the ODP-RM. It is expected that the two architectures will merge at some future date.

The OSA architecture is presented from two perspectives; OSA(app) and OSA(sys). OSA(app) views the architecture from an application perspective, i.e. service analysis and design, whilst OSA(sys) views the architecture from a system perspective, i.e. for a constructor of an environment on which the services can be provided. OSA(sys) acknowledges that there will be a move away from 'exciting' technical solutions to ones driven by the market and business developments. OSA(sys) is a distinct architecture. It provides a network model made up of ODP-RM mechanisms running on logical nodes and deals with platforms on which services may be implemented. The system designer is concerned with the system (service machine) providing the service.

A service machine is a distributed system of functions, and supports management and control (to glue the elements!); it turns a system perspective into a service perspective. In ODP-RM terminology the service machine can be thought of as 'middleware'. Middleware has two views: one faces towards the technology and is application independent, being supplied by libraries, and is responsible for binding, authentication and location, etc.; the other faces towards the application and is application specific, and is constructed using system tools being responsible for transactions, replication, persistence, transparency, etc. OSA(sys) is restricted by the limited capabilities of the lower level resource, possibly stemming from physical limitations. The OSA architecture is considered a high level platform which sits on an infrastructure seen as functional units. The OSA(sys) uses an ODP-RM approach and models the network as a series of interconnected service nodes which are a representation of resource access and adaptation. Each service node comprises a nucleus object and resource adapter, distributed processing and deployed computing objects.

4: The ISE Platform

A platform is a realisation of an architectural style in a form that provides a precise functionality. The architecture constrains the designer to engrain the style into any realisation. A platform realisation can be said to conform to the architecture as viewed by the engineering viewpoint, i.e., the deployment mechanisms (the resource units - node and capsule - and the distribution unit - cluster) as well as communication concepts (transport networks and kernel transport network channels) and the basis of system management. The designer of the platform will use architectural components to provide the transformation from requirements into a product, and to do this the designer needs to be aware of how the various building blocks are realised by the technology. Within the platform development there lies both a methodology and a formal method for the transformation.

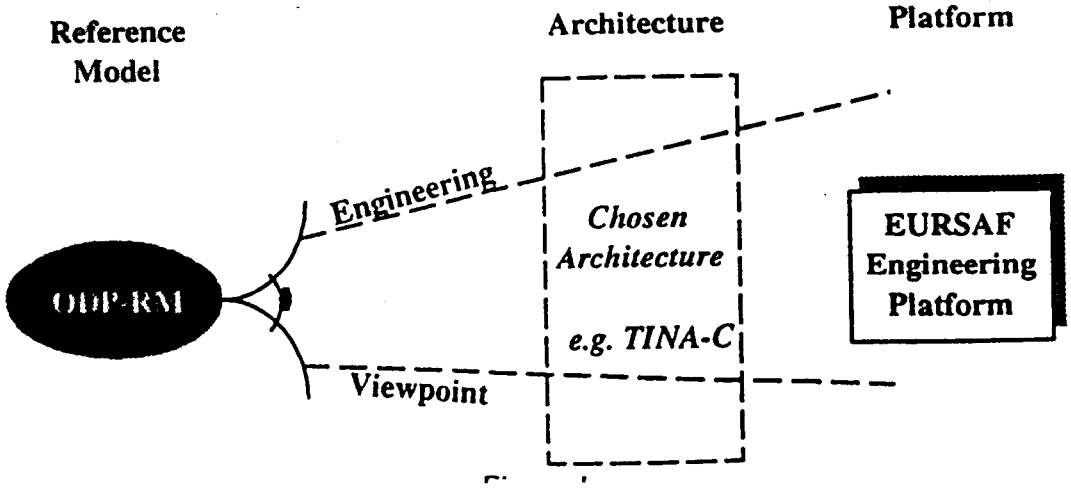
The ISE engineering platform details the mechanisms required for the transparent interaction of objects that comprise a telecommunication service. It is shown in outline in figure one, and comprises a distributed environment and application development environment. It is the infrastructure that provides the distribution transparencies and communication facilities between objects, independent of the local operating system and the heterogeneous underlying technology. It consists of a kernel, or nucleus, and a collection of servers. Examples of the servers include:

communications
version control
management
trader
security.

The application development environment is a set of tools for the development of applications for the distributed processing environment, e.g. stub generator, and includes:

protocols
stubs
bindings
transparencies
generic run time
etc.

The ISE platform focuses on the engineering specification of the deployment of objects in order to execute them. The engineering specification will focus on the particular requirements of telecommunications, addressing such topics as the ability of this environment to support real time interactions, scaleability for supporting large to very large service systems, manageability and migration. The latter, within a distributed processing environment unstudied in other consortia, will provide insight on how existing service provision systems can migrate to a platform for ISE.



This focusing should allow the definition, in general, of a unified interface of the building blocks [objects, components or clusters] that comprise the platform and an association mechanism suited to the requirements of telecommunications.

5: Conclusions

The design of an engineering platform for a wide range of different services that can be integrated and controlled within an open distributed environment is very complex.

There is a need to provide many engineering characteristics in a transparent and manageable fashion. The platform must support transparency to the underlying technology in the majority of actions, but must allow some translucency to allow the services to be managed by the service provider and/or user. An extended object oriented approach can provide full flexibility to meet these requirements. The need to select a suitable architecture is essential to ensure efficiency and inter-working standardisation.

References

1. Linington, P.F., "Introduction to the Open Distributed Processing Basic Reference Model". Elsevier Science Publishers, 1992.

2. Magedanz, T., "IN and TMN: the basis for future information networking architectures". Computer Communications, Vol. 16, May 1993.

3. Sharp, C.D., "Advanced Intelligent Networks - Now a reality". 4th IEE Conference on Telecommunication, April 1993.

4. "ROSA Architecture", CEC ROSA, workpackage Y, Deliverable 93/BT/DNR/DS/A.005/b1.

5. "Integrated Service Engineering - CASSIOPEIA", CEC RACE project R2049. Deliverable R2049/CRA/SAR/DS/P/014/b1.

6. "European Standards Architecture Formulation", (EURSAF). CEC RACE project R2124.

7. Fuente, L., "Application of TINA Architecture to Management Services", Proc. Int. conf. of Intelligence in Broadband Services and Networks, Aachen, September 1994.