Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Architecture and Frameworks

David Iggulden, Owen Rees, Rob van der Linden

Abstract

This document puts the old frameworks document RC337 together with T10 and T11 deliverable (RC.420 and RC.430) which are concerned with explanations and placement of the concepts of the Enterprise and Information paradigms for Open Distributed Processing.

ANSA is described as an architecture plus a set of related frameworks, each suitable for a particular purpose. It separates the cognitive framework and projections from the architecture proper and shows a way in which issues of conformance to the architecture and the frameworks can be examined.

Purposeful and meaniful interaction are the basis of the set of Enterprise and Information concepts described in the report. These concepts relate to both those used in describing and specifying the infrastructure supporting a distributed application, and the design and development of applications.

.

APM.1017.01 **Approved** 15 February 1994

Technical Report

Distribution:
Supersedes:
Superseded by:

Architecture and Frameworks

Architecture and Frameworks

David Iggulden, Owen Rees, Rob van der Linden

APM.1017.01

15 February 1994

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK	(0223) 323010
INTERNATIONAL	+44 223 323010
FAX	+44 223 359779
E-MAIL	apm@ansa.co.uk

Contents

1 Introduction

1.1 Audience

Distributed processing, as a discipline, is now at a point where the effects of technological change and its implications for ways of working can be considered at all levels. The particular audience for this document is:

1. Those interested in the design process and the principles of architecture
2. The developers of standards frameworks
3. Those whose concerns are with the relationship or “placement” of standards and technology.

1.2 Scope

This document provides a background to the explanation, validation and placement within the ANSA Architecture, of the Enterprise and Information concepts and uses the idea of frameworks.

A framework, in this context, is a structure into which various architectural concerns can be placed and related to one another.

The frameworks behind the ANSA architecture consist of:

- a cognitive framework,
- a model-theoretic framework,
- a constituency framework,
- a design process framework,
- a tools framework,
- a quality framework.

An examination of the cognitive aspects of system analysis and design - the cognitive framework -, together with a consideration of complexity, leads to the idea of working with a number of projections on a system. A projection, in this context, is a total view of a system using a consistent set of abstractions.

For the problem domain of Open Distributed Processing (ODP), five projections - the model theoretic framework - have been found to be useful:

1. enterprise
2. information
3. computation
4. engineering
5. technology.

The constituency and model theoretic frameworks provide partitionings of the problem and solution space. The constituency framework identifies the

various stakeholders in the analysis, design and operation of a distributed system - the problem space. The model theoretic framework provides a distributed systems architecture - the solution space.

The design and tool frameworks serve particular purposes related to model-building and software development. They support the processes of enterprise and information modelling which are necessary for the development of large complex systems.

Finally the frameworks are provided with an overall umbrella by a Quality Framework where Total Quality issues and detailed notions of Quality of Service parameters (including reliability, security, availability, timeliness, etc.) are considered. These become a pervasive concern when dealing with distributed autonomous systems.

Each of these frameworks is discussed in greater detail in chapters 2 - 4. Enterprise concepts are explored in chapters 5 - 7 and information concepts in chapters 8 - 12. The relationship of enterprise and information with the other projections is outlined in chapter 13.

2 Frameworks and Architecture

2.1 Frameworks

Activities related to the development of computing systems can be initially distinguished into:

- concepts, theories and models
- specifications constructed according to these theories
- modelling processes - that is, the methods, tools and techniques used in the construction of the specifications and the implementation of systems based on the specifications.

This distinction leads to the idea of a number of frameworks in which practical considerations of the design and implementation of large complex systems are placed relative to one another.

Design can be viewed as a series of transformations. Verification and validation are the means used to check that the transformations are appropriate. Verification checks that information has been preserved and validation checks that the transformed system is fit for purpose. These ideas of verification and validation are used to relate the broad areas of requirements, design, development and support. Non-operational requirements are viewed as providing a suitable means of support to the validation process.

Design also takes into account complexity as well as distribution. Complexity includes notions of volume and scale as well as difficulty, and methods and tools used currently to deal with these problems need to be re-considered in the light of new problem characteristics where a number of former assumptions may no longer be found to be tenable in distributed systems. New methods such as object-oriented design need to be evaluated within the design and tools frameworks and their relationship with the underlying basis provided by the ANSA architecture explored.

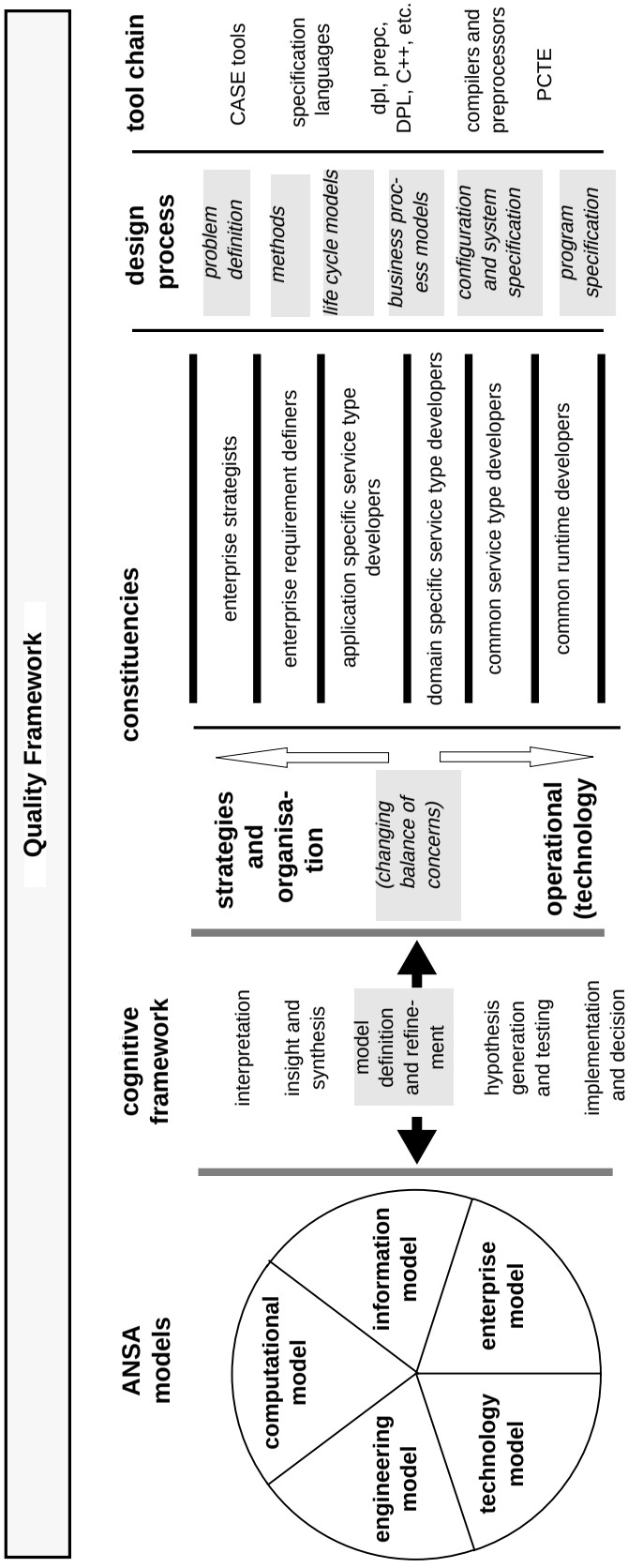
A number of fundamental notions such as entity, structure, composition and behaviour are used throughout the frameworks; the frameworks as a whole are considered in terms of the necessary relationships between those notions and their animation in the context of design and development systems.

An overview of the frameworks applicable to design and development is illustrated in figure 2.1. The cognitive framework leads, on the one hand, to the ANSA models or model theoretic framework and, on the other, to constituencies, which in turn lead to the design and tools frameworks. The rest of the document provides support and justification for this structure.

The approach taken to distinguishing the entities and concepts for the Enterprise and Information aspects of ANSA can be briefly summarised as follows:

- Enterprise is concerned with **purposeful interaction** characterised by **conversations** between **agents**

Figure 2.1: ANSA Architecture - Frameworks



- The parties to this interaction, who may be individuals or organisations of individuals, will need to arrange the terms of the interaction, unless constrained by some super-authority;
- The interaction will therefore be concerned with negotiating the exchange or availability of abstract commodities (or references to concrete commodities). In this context information is considered to be a commodity;
- The agreements reached or acknowledged by these negotiations will be represented by particular instruments known as contracts which consist of references to promises, penalties for broken promises and to the process of arbitration to resolve disputes;
- The negotiating process, and the process of ensuring that the terms of contracts are fulfilled, are the concern of management.

Broader statements of intent, with less emphasis on interaction and negotiation, are called policies and play a large part in the specification of distributed systems.

2.2 Background to the architecture

The ANSA architecture applies to a class of computing systems in which different parts of a distributed application work together in a unified and natural way. Application interworking can be achieved with a relatively small effort by the application designers, who do not need to concern themselves with the specialised diversity and physical distribution of different computers, operating systems and network protocols which comprise the technological base of the underlying computing environment. Instead, tools are provided to hide many of the complexities, normally encountered in the design and implementation of distributed systems. By removing the burden of technical detail from the designers, they can instead concentrate on meeting the requirements of end users and on issues concerning the provision of applications which span organisational, political, and geographical boundaries. The ANSA models and the associated frameworks provide design guidance in this area as well.

Once the technological boundary that hinders application interworking in heterogeneous computing environments is removed, there will be few limits on the extent to which users can combine their computer systems, applications and data. This will lead to very large open distributed processing systems. For such systems to become and remain effective, they must satisfy several demanding requirements. Specifically, they must:

1. allow many different kinds of applications, including office, factory, telecommunications, and general data processing applications;
2. allow many different technologies, such as hardware platforms, operating systems, communications protocols, and programming languages;
3. scale from small (a few nodes) to very large (the size of the existing telephone network and larger);
4. allow applications to span organisational boundaries, whilst satisfying the demands of all the organisations they span;
5. allow graceful evolution, so that requirements can continue to be met without green field developments, which will become economically unjustifiable;

6. allow the inclusion of the considerable installed base of computing hardware and software, the so-called legacy systems, thus protecting existing investments;
7. offer opportunities for many hardware and software vendors, so that new innovative products will continue to fit with the installed base, extending it where necessary without the introduction of discontinuities.

The scope of an architecture that allows systems to be developed with all these characteristics is too large to be usefully captured by a model based on a single consideration or abstraction. For instance, trying to provide a mathematical representation of issues relevant to the architecture would lead to a model which would be difficult to comprehend and would therefore be useless to most people. Similarly, merely showing the architecture as a set of interrelated tools would lead to an incomplete picture, useful to some but not to others.

During the development of ANSA, it has become clear that the only way to tackle this dichotomy between the two extremes is to provide for separation of concerns. A convenient way to give context to the work which is going on under the ANSA umbrella is to portray it as the development of a set of separate, but related frameworks. There are no rules for the choice of the particular set of frameworks chosen for ANSA; there may be different sets of frameworks that could be just as effective. The ultimate test for a suitable choice is whether those that are meant to use the frameworks agree they are a workable set. ANSA sets out the following frameworks:

1. *cognitive framework*: five cognitive processes of design and problem solving from which the five projections have been derived. These cognitive processes also implicitly underlie the approach to activities in all the frameworks;
2. *model theoretic framework*: five models, with concepts, rules, recipes and guidelines. This forms the keystone to the architecture;
3. *constituency framework*: a framework partitioning tool users, such as designers, programmers, users, maintenance personnel, and their concerns as well as those of problem owners (and their problems), system owners, vendors etc.;
4. *design process framework*: this framework contains notions of methods and life-cycle models as well as elicitation, specification and model description languages. It has been labelled Design Process in figure 2.1 to indicate the importance of design but languages are also an important feature of both this and the tools framework;
5. *tool oriented framework*: elicitation, specification and programming tools, including preprocessors, transformers, compilers, linkers and configuration tools;
6. *quality framework*: this provides for the statement of non-operational and security requirements and for fitness-for-purpose considerations.

Moreover, this set, especially the model theoretic framework, has contributed to the work in ISO on Open Distributed Processing and as means of relating industry initiatives such as DCE and CORBA. The frameworks thus provide a basis for achieving consensus between these activities.

3 The frameworks

3.1 The cognitive framework

The following classes of cognitive processes of design can be distinguished:

1. activities to **interpret** each system owner's concerns and the rules which a system is desired to observe;
2. operations of **insight** and **synthesis**: activities to determine what system features give rise to particular concerns and what kind of models are required to represent those concerns;
3. operations of **model definition** and **refinement**: activities to determine the system operations that represent real world activities, both legitimate and illegitimate;
4. operations of **hypothesis generation** and testing in exploring the consequences of a particular system description: this includes the real world interpretation of the operations and the limitations on their scope;
5. operations of **implementation decision**: deciding how system features may be represented in existing technology and how existing technology may be modified to fit a particular system design.

The cognitive framework was derived from a philosophical and psychological analysis of the various cognitive processes that may be distinguished when problem solving and making decisions. A framework resulting from the classification of these processes is not unique or original: it is the foundation of Piaget's concept of intelligence [PIAGET78] and was developed by Humphreys in connection with a theory of decision making in the face of uncertainty [HUMPHREYS84].

It was recognised that these processes are by no means ordered. They sometimes happen simultaneously, sometimes in a particular order, and both situations can arise during the design of a particular system.

Of all the process classes above, the operations of insight and synthesis apply most strongly to the development of an architecture. These operations must lead to the development of a system of models and notation to describe substantive features of the system that is being designed. As most systems are very complex, a single model and calculus is unlikely to suffice. The complexity we are considering comes in two varieties - that of difficulty and that of scale.

A common way to cope with complexity is to separate concerns. This leads to the identification of a set of paradigms or projections, each of which can be explored separately.

An examination of the field of distributed computing, through a study of current research in distributed computer and system design techniques, yields the identification of five projections:

1. **enterprise projection**: the purpose of which is to explain and justify the rôle of the computer system within the organisation, The enterprise

projection with its concern of requirements and purpose generally extends in its coverage to systems wider than those that are computer-based.

2. **information projection**: the purpose of which is the identification and location of information and the description of information processing activities;
3. **computational projection**: the purpose of which is to help structure programs for modularity and parallelism, for linking separate applications into integrated packages and for making programs independent of the technology on which they run;
4. **engineering projection**: the purpose of which is to allow reasoning about performance, resource use, and system configurations;
5. **technology projection**: the purpose of which is to provide blueprints of systems during their construction and maintenance in terms of their physically recognisable components.

This describes the way the projections can be used to outline the concerns expressed in the design and development of a computer system. More abstract notions of separated and definable concerns which map to these five projections are ideas of purpose and obligation, notions of reference and identity, causality, as well as the choice of supporting mechanisms and their configuration, and conformance. Existing or proposed systems can be described in any one of these projections, but each of the projections has a different purpose, and so has the resulting description. There is a sense in which the descriptions taken together complete the description of a system. This entails the idea of suitable languages used for this purpose with sets of terms which embody the separate areas of concerns.

The projections express separate concerns. When applied to a particular problem domain they lead to a set of models and to languages for describing the models. The problem domain here is that of networked systems. The application of the projections leads to the model theoretic framework of the ANSA Architecture.

3.2 The model theoretic framework

ANSA is concerned with the description of an architecture, that is a set of facts and rules that need to be taken into account when designing and building distributed computer systems. The application of the projections, identified in the cognitive framework, to the set of facts and rules yields a set of models in the model theoretic framework. The five models together define the architecture.

Each model provides for consistent use of terminology, identifying important concepts and components that need to be represented in descriptions of distributed computer systems. A set of rules constrains the way in which such components can be combined in system designs that conform to the architecture. A set of recipes provides advice on how to combine basic components to obtain certain useful components. Recipes are not mandatory, but using them will ease the solution of any interworking problems that may arise later on. Finally guidelines are provided to help designers make design decisions if their own preferences do not offer sufficient basis for such decisions. The recipes and guidelines are provided in the particular prescriptions offered in the design and tools frameworks.

The purpose for each descriptive model is to help designers of distributed systems make sensible design decisions. The abstract nature of the model theoretic framework makes it less effective for everyday use. The designers of ANSA use the model theoretic framework as a basis for the definition of languages and tools that relate more directly to the problems that distributed systems designers face (see the next section).

The model theoretic framework consists of five models, each offering a set concepts and rules:

1. *enterprise model*: which allows the construction of
   (i) a model of an organisation and the changes in it that a system must assist, or induce, and
   (ii) a model of a system in such a way that it is possible to reason about the extent to which it will in fact assist or induce those changes;
2. *information model*: in which the concepts allows the designer to
   (i) model the use of information as the representation of obligations that exist between agents in an enterprise and as an instrument which mediates those obligations,
   (ii) describe the components of a particular system in information terms;
3. *computational model*: which provides concepts of invocation and structure that allows
   (i) the definition of the facilities required of a programming system that permits the implementation of distributed applications for an ANSA infrastructure, and
   (ii) reduces as much as possible the view that a programmer requires to have of the detail of the ANSA infrastructure;
4. *engineering model*: which defines the functions of an ANSA infrastructure;
5. *technology model*: which allows a statement of the conformance rules on the set of components which provide the realisation of a programming environment and the ANSA infrastructure.

In Part 3 of the ISO Reference Model of Open Distributed Processing[ISO92b] an equivalent set of models are referred to as languages.

The computational model has been completed and agreed in advance of any of the other models. It is described in [APM.1001.1 91] and extensions to it are identified in [APM.1004.1 92]. Further clarification and extension of this model is documented in reports on Trading and ANSA's Security Model.

The engineering model proves to be very complex, both in extent, and in terms of the multitude of options that can be used to provide selective distribution transparencies through an ANSA infrastructure. Consequently, the problem is tackled in steps, one subject area at the time. The report on Interface Groups [APM.1002.1 91] provides one abstraction of communication, synchronisation and error recovery mechanisms, harnessed so as to provide increased performance and robustness. The report on the Atomic Activity Model and Infrastructure [APM.1004.1 92] examines the requirements of a class of robust computations, called atomic activities. These include assumptions that can be made about the physical environmental failures and statements about the required semantics of the mechanisms essential to the control of atomic activities. The report on Addressing defines a number of requirements on addressing, location, and migration mechanisms.

The enterprise model is concerned with the identification and positioning of abstract notions of agents, activities and resources and their relationships as well as issues of authority, policy, contract etc.

The approach to the information model extends the usual notions of information structure to consider the problem of naming in federated systems and the exchange of information via conversations. The ANSA Naming Model[APM.1003.1 92] defines a naming model that permits the discussion of federation of separately administered naming systems and is part of the information model.

3.3 The constituency framework

The constituency framework is concerned with identifying those that use ANSA, its architecture, its languages and design methods, and its tools. The structure of this framework is essentially the same as that proposed in [ARM89] which contained the following constituencies:

1. non-users: such as system owners and vendors;
2. system designers: those that design the distributed system infrastructure;
3. designers of the tool environment, used by the service providers;
4. service providers: application designers and programmers;
5. service users: those that use the distributed applications;
6. system engineers: those that maintain and manage the distributed system with its services.

The audience for this document identifies most strongly with **service providers.** These are the “middleware providers” who tailor distribution mechanisms to application needs.

This framework is used to identify target audiences for particular aspects of the architecture, languages, and tools. The categories should be used as indications only: it is not expected that everyone who is involved with a distributed computer based system will always fit in one and only one category or that the categories map into only one projection. In other words the constituency model is concerned with identifying a partitioning of real users who will have some functional rôle or agency in an identifiable domain.

The notion of constituency and identifiable domain also extends to the characteristic problems encountered within that domain. This allows us to identify generic problems and their solution.

3.4 The design process framework

The Design Process framework illustrates and explains the consequences of making choices for particular methods and languages. The components of this framework will be more familiar to designers and programmers than those of the model theoretic framework. As the architecture only becomes visible to the public at large through implementation, this framework and that of the tool chain are of the utmost importance in justifying the work on the architecture in the abstract model theoretic framework. ANSAware [AIM92] is a very important exponent of the tool and design frameworks (and as such an example and existence proof of the Architecture).

Methods and life cycle models are essential for the management of large systems where some division of labour is mandated by sheer size. The methods currently available do not explicitly tackle distribution. It is the concern of the Enterprise and Information projections to provide enough concepts for the development and the extension of existing methods.

The development of tools and languages for the construction of description of systems in enterprise and information terms cannot begin until it becomes clear what precisely are the requirements on the semantics of such languages. Since the enterprise and information models that define these semantics have not been completed, the development of languages and tools will have to wait. Nevertheless, as part of the development of the Enterprise and Information Models, it is worthwhile to build up an understanding of existing languages and tools that claim to provide solutions to the design problem. This work is currently in progress.

Since programmers are generally concerned with computational as well as engineering issues, existing programming languages reflect both concerns, but without a clear separation. Most existing programming languages do not support distribution either. A number of assumptions that can be made in centralised systems (and that lead to simplifications of mechanism) can no longer be made in distributed systems [APM.1020.1 93]. The avoidance of such assumptions has a profound impact on any programming language and its associated tools.

3.5 Tools framework

The tools framework is intimately concerned with instrumentation for the methods and languages in the design process framework. The idea of considering them separately was because they are distributed systems in their own right and are, therefore, a prime candidate for applying ANSA architectural principles. Some of the tools mentioned in figure 2.1 are already part of the ANSA armoury, others, such as CASE tools which act as a respository for methods and their animation are being reviewed in order to assess the impact of distribution. In some tool sets language translators are included which produce implementable solutions starting from a statement of requirements. This requires an architectural view of the transformations and verification involved and provides a clearer argument for separating out the tools framework as a distinguished component of the ANSA architecture.

3.5.1 Languages

The languages and supporting tools that are proposed by ANSA have been developed with the model theoretic framework in the background, thus providing a clear separation between computational and engineering issues. Obviously, there are many ways in which programming languages can be provided, and to date ANSA has suggested two general approaches. These provide two syntactic options which allow the new semantics of distribution to be expressed.

The first approach is to extend an existing programming language. In ANSAware, such extensions are pre-processed and converted back into the native programming language syntax, making use of library calls where possible and appropriate. In this way, no changes to the original programming language and compiler technology are required. Care must be taken to ensure

that the use of the original programming language does not violate any distribution requirements, since few guards are provided against doing so.

This leads to the second approach to providing a suitable programming language, adopted by ANSA. It is to provide a new programming language to the designer and to restrict the semantics of that language to just that which can be properly defined in a distributed system. ANSA includes a proposal for Distributed Processing Language (DPL), an example of a language whose semantics are strongly linked to the ANSA Computational Model (in the model theoretic framework).

In order to protect investment in existing code, DPL can be used to effectively wrap up existing code components, making the services they offer available in a distributed environment. This also provides a useful migration path from the pre-processing approach to the second approach that includes DPL.

In other systems, such as Amadeus [COMANDOS], extensions and alterations to the syntax of the programming language itself lead to changes in the compiler, thus effectively creating a new (but very similar) programming language. The problem with this approach is that if the “new” language is to cater for distribution, then many of the “old” semantics are no longer valid. Because the new and old language are still very close to one another from a syntactic point of view, confusions can easily arise. If the language is to change, then why not go all the way?

3.6 Using the frameworks

The use of the model theoretic framework of the ANSA Architecture together with the development frameworks will allow an orderly comparison of different solutions to distributed programming and design. Initiatives such as ANSAware [AIM92], OSF/DCE [OSF91] and OMG/CORBA[OMG91], for instance, provide a necessary basis for interworking. In particular, the model theoretic framework will aid the analysis of different distributed system solutions, so it becomes obvious what will need to be done to get an OSF/DCE based system to interwork with an OMG/CORBA or an ANSAware system. (It may still be difficult to provide interworking, but at least that will be known before attempting to provide it.) The model theoretic framework, and especially the engineering model, is also a tool in the analysis of the significance of market trends with respect to available products and product development strategies.

The frameworks can be used in the analysis of what needs to be done to achieve portability across different distributed systems solutions. In that sense the architecture is of real assistance to the designer who will increasingly be faced with multiple solutions to the distributed processing problem, none of which have all the required characteristics, and few of which will actually interwork with any or all of the “other solutions”.

Note that the frameworks are structured independently and in particular, that there is no necessity for one-to-one relationships between components in each framework. It should be clear that programmers will take the purpose of the system they program (expressed in enterprise model terms) into consideration whilst using programming languages that have been developed with some computational model in mind which deals specifically with the difficulties and opportunities provided by distribution.

3.7 Background to ANSA and standards

It is clear that deriving an architecture that allows the design and construction of distributed systems requires a consensus amongst the major players in the computing and telecommunications industries and ANSA has provided a forum for that consensus by being directly supported by more than 20 companies in these industries. The development of ANSA has as a consequence also maintained close association with a number of industrial and consensual standards-making bodies. On the industrial side there are strong links with the OSF, UI and OMG initiatives and on the consensual side with the work on the Open Distributed Processing Reference Model standard which is being developed jointly by the ISO/IEC Joint Technical Committee One (International Standards Organization/International Electrotechnical Commission) and the CCITT (The International Telegraph and Telephone Consultative Committee).

In the standards context, ANSA forms a testing ground in which architectural concepts are identified and verified. Verification is done by prototyping. Its close association with the international standards making process allows ANSA to adjust its concepts and aims to meet further requirements identified by the ODP community and also to influence how those requirements are met. ANSA's general notion of objects is already aligned with the Reference Model for Open Distributed Processing Part 2 Descriptive Model which is at the Committee Draft stage [ISO92a] and the ANSA view on the architecture of the engineering model is strongly reflected in the committee draft of the Part 3 Prescriptive Model [ISO92b]. Several of the companies supporting ANSA are also active in both types of standards making and advantage is taken of both formal and informal links to obtain a high profile for the consortium's views.

4 The enterprise projection and model

4.1 Introduction

The enterprise projection and model is concerned with reviewing the minimum set of concepts and validating their applicability to the development of complex information processing systems. The task also subsumes what is meant by compliance to an enterprise architecture and how it can be defined.

The enterprise architecture:

- defines a set of terms which enable the purpose of an Information Processing (IP) system and that of the wider system of which it is a part to be represented and reasoned about;
- enumerates a minimum set of useful concepts and their relationship to enterprise models and enterprise modelling.

The purpose subsumes notions of activity so that a useful characterisation of enterprise architecture is that of providing a structure allowing reasoning about purposeful activity.

The ODP Reference Model uses enterprise concepts and terms to define a language used in the complete description of an ODP function [ISO92b]. These concepts and terms are related to those which this document discusses at greater depth. Not all of the ideas expressed here have been presented to the ODP community.

The statement about the IP system and the wider system of which it is a part supposes that the boundary within which the IP system is defined is an obvious property of that system, which is not always the case. The idea of a boundary, therefore, is an example of a basic concept which needs definition and explanation.

The chapter introduces three main sets of ideas:

1. concepts of **interaction** based on the notion of **conversation**
2. concepts of **structure**
3. **taxonomies** classifying those associated with specification, development, implementation and use of an IP system

The sets of basic concepts and taxonomies are expected to be particularly useful in helping designers make decisions about the way in which a system or application should be structured (distributed) such that the resulting system or application fits the environment in which it will be used. The enterprise model of a particular enterprise can thus be seen as providing a context in which a distributed system is designed. The construction concepts provide a framework for introducing process concepts and notions of contract and policy.

Current IP systems are dependent on the use of shared data. In distributed systems, especially federated systems, where separate autonomous systems are combined then that notion of shared data no longer applies. The successful combination therefore needs negotiation and interaction. Conversations

provide a model of interaction and this concept is crucial to the whole architecture.

This chapter provides the background and the particular context within which enterprise architecture is discussed. It makes the implicit point that there is a range of such architectures of equal validity but requiring clear statements of context. Chapter 5 is about the set of concepts and chapter 6 indicates some process constructs that could be used to develop methods of system construction. Chapter 7 summarizes the concepts of the enterprise architecture.

There are a number of analysis and design methods and work is in hand to establish a methodology and coherence to this field [OMG-OAD92]. This document contributes to this debate and provides a basis for positioning and evaluating tools and methods. More specifically it contributes to understanding the impact of distribution issues on tools and methods.

4.2 Audience

This chapter is directed to those concerned with large scale application design and who wish to see how business requirements relate to technical design issues especially those of distribution.

4.3 Organisational needs

It is recognised that a major problem associated with the design of distributed systems that serve some organisational need, is that the structure of the IP system cannot be separated from the structure of the organisation it serves. There is thus a need for a set of concepts which enable the organisational environment of a system to be thought about as carefully and rigorously as the IP system itself. These concepts must be sufficiently generic to be capable of satisfying two sets of needs: to model an organisation and the changes in it that the IP system must assist (or will induce); and to model a system in such a way that it is possible to reason about the extent to which it will in fact assist or induce those changes.

4.4 Enterprise models and enterprise modelling

“Enterprise” is thought of in a number of ways. Two of the most common refer to either an activity or to an organisation. Business or firm may also be used as a synonym for enterprise (used in the sense of organisation) although the term business shares in the same double connotation as the word enterprise. Enterprise also has a connotation of value as, for example, in the slogan ‘DTI: the Department of Enterprise’.

“Models” also has a similar usage problem. In the model theoretic framework the enterprise model is an interpretation of the terms of the enterprise language used to express the enterprise concepts. Model is also used in the sense of an exemplar or standard, or as some suitable abstraction of the real world which can be reasoned about and manipulated and which may possibly be executed as part of the real world.

"Enterprise Model" will be used here in the latter sense; that is, as an abstraction of the whole or part of an enterprise or organisational unit rather than the set of enterprise concepts.

An organisational unit is that part (or whole) of an enterprise for which a boundary and a purpose can be defined. Used in this sense it then follows that enterprise modelling is a process which has as a result an enterprise model. Both the model and the process will be informed by the enterprise concepts: so that for example, the ideas of boundary and purpose are enterprise concepts.

The purpose of this paper is to review, define and explain the useful set of concepts used to build and reason about enterprise models and the processes used to develop them[1].

The need to relate the enterprise architecture to the computational and engineering models and also to explicitly relate the work of enterprise and information modelling to the architecture has been noted in the chapter on Architecture and Frameworks (chapter 2). This considers a technical framework in which the modelling work and the design process would fit, as well as the basic models, in order that the relationships are correctly placed and understood. It allows the ideas of methods, life cycle models and the use of tools from the tools framework, such as CASE, to be considered from the point of view of distributed systems.

In summary the main message is that modelling allows us to understand the technology and how to best deploy it.

The next section discusses the background, nature and opportunities of distributed processing applications.

4.5 Background to distributed processing

4.5.1 An approach to understanding distribution

It is clear that if an architecture is to form the basis for a particular design, the designer is required to assure himself of the fact that the Architecture is not ruling out the achievement of "the proper balance between human, organisational and technological issues". For the architecture to be considered by designers at all, it will be necessary to show explicitly that it is an appropriate vehicle for a design.

Hitherto in ANSA an implicit understanding of the nontechnical issues impinging on the design of distributed systems and application programs has been relied on. To make those issues explicit it will be necessary to identify, express and systematise a number of concepts such as organisational

1. One terminological problem is that the modelling activity focused on the development of information processing systems for an enterprise is variously referred to as "enterprise modelling" or "information modelling". The view taken here is that "information modelling" is a species of enterprise modelling that is more concerned with defining information and information flows and that the phrase "information model" is used here as a surrogate for information concepts such as structure, naming, and reference [chapter8]. Therefore, information is viewed here as a commodity or artefact of the enterprise and the process of information modelling has as a result an enterprise model of a particular sort [DRETSKE81].

structures, ownership, responsibility, obligation, permission, information structures and control structures and place them in a proper architectural framework. Distribution, especially when heterogeneous systems are involved, brings such considerations to the forefront.

The approach taken here allies itself naturally with the techniques of the 'soft' systems methodologies [CHECKLAND81] and with the concerns of analysts of sociotechnical systems and other approaches [LYYTINEN87].

To identify and systematise the essentially nontechnical issues, it is necessary to understand the areas in which such issues are studied. Hence such topics as organisational theory, the theory of law and legal systems, economics, business, politics, and human issues need to be considered. Studying these areas is necessary in order to understand how ANSA systems fit in environments where such issues are of paramount importance. It is necessary, however, to guard against making such study an end in itself.

The central point in the investigation of nontechnical issues must remain with the primary exponents of the Architecture: the tools (preruntime) and the distributed system environment (runtime) that must satisfy the concerns of all those involved in the distributed system: application designers as well as users and system owners. Only when all identifiable concerns that can have a bearing on the technological issues have been taken into account, can we claim to possess an Architecture that we are confident will form the basis of many successful designs and implementations.

The effect of distribution on the development of IP systems is such that a number of assumptions which are implicit in centralised systems no longer hold[APM.1020.1 93]. Because of this models of computation play a central part in architecture and are a prerequisite to understanding the approach taken here. The ANSA Computational Model is described in [APM.1001.1 91].

4.5.1.1 Dealing with complexity

In designing large, distributed, heterogeneous systems we need to deal with complexities of two kinds. The first is concerned with sheer size where there are large numbers of similar items and the second that of difficulty where the difficulties may be removed by adopting newer paradigms and abstractions.

ANSA does this by separating the concerns surrounding the problem. This done by partitioning the issues and participants in a logical fashion and also by providing suitable abstractions. The first partition can be thought of as meeting the difficulties caused by sheer size, and the abstractions provide a way of tackling the other kind of complexity. This approach is introduced in chapter 3.

4.5.2 An approach to solving the problem

4.5.2.1 The design process

A system needs to be designed. A design process is generally imposed to both streamline the activities and to allow the measurement of progress. The design process will in all likelihood embody a method which it may be claimed will ensure certain properties of the developed system.

The ANSA architecture identifies an abstract design process, based on the canonical design step. It is deliberately abstract, so as not to include any hidden judgements about the value of proceeding in this way or that.

The canonical design step takes a description of a system (a design) and transforms this into another description. The transformation is generally a mechanical process (whether automatic or manual). Each step is sufficiently small to give confidence it will succeed. The new description will be verified against the old description to ensure that the result of the transformation is internally consistent and that it displays all the properties required by the description from which it was derived. Verification is essentially a mechanical process.

Validation is the process that ensures that the old and new descriptions are fit for their purpose. This is essentially a human activity, that is, it requires value judgements.

Any particular design process will be tailored to fit a particular set of objectives, and can thus be seen to contain implicit value judgements. ANSA does not depend on any one particular design process, as it is recognized that the choice of design process is a matter for the design enterprise. The design process is thus not an intrinsic part of the architecture.

4.5.2.2 Modelling

If a system is to be designed by several people, then they must all be able to talk about the design. It must therefore be possible to express it in some form. The description of a design is often called a model; the process of obtaining such a description is known as modelling. In abstract terms, modelling requires a choice for a language. This implies a syntax (a notation, a set of symbols), a semantics (the meaning that should be attached to the symbols) and a pragmatics which captures those aspects of meaning that are not accounted for by the semantics. The languages chosen will be embodied in the design methods used and in the tools used to support these methods.

ANSA does not (in principle) insist on particular syntax, but does require the expression of semantics, so its concepts may be defined. From time to time ANSA must select a particular syntax in order to express its concepts. Such a selection should not be construed as a part of the architecture although the structure of the frameworks requires the consistency of any set chosen.

ANSA does not make mandatory the use of any particular formal methods or description techniques, or of semiformal methods such as Data Flow Diagrams, Entity Attribute Relationship or State Transition Diagrams. It is most likely that methods embodying object-oriented techniques i.e. those being studied by the OMG's Analysis and Design SIG [OMG-OAD92] will play a large part in the methods adopted for the design framework.

4.5.3 Conclusion

The issues of design processes and modelling must be seen as ancillary to the architecture. The main thrust in the design of the architecture must be in the population of the projections. It is important that the enterprise architecture will be accepted in a wide forum as a basis for understanding this issue, so the relationships with current system design practice (design process and existing modelling techniques) must not be ignored altogether.

4.6 Fitness for purpose (of architecture and designed usable system)

There is a general realisation amongst designers, and not just designers of distributed systems, that the quality of a system is judged by how well it fits

into its environment and how easy it is to tailor that system to suit the changes in the environment and the underlying hardware and software technologies that support it as well as views of on its reliability and performance.

Judgements about the quality of systems are essentially a personal matter and it is the designer's responsibility to achieve a proper balance between human, organisational and technological issues although metrics must also be considered.

The overall quality requirement supposes the convergence of the metrics associated with each of the quality parameters involved and of the soundness and completeness of the accompanying functional operations.

The processes of verification and validation therefore need to be a central part of the methodology of development. The design and development process itself should be carried out within a total quality framework using available standardised procedures and certification. Validation of a system, when formally included in this process, provides the necessary bridge between the various parties involved in the ownership, definition and procurement of distributed processing solutions (chapter 3).

The problem for which we seek a solution may perhaps be best summarised by the question:

- How do we design, build, operate and maintain distributed computer systems that comprise a wide variety of hardware, operating systems and communications protocols, that scale from small (a few nodes) to very large (millions of nodes), that support many different applications (Office, CIM, MIS, Telecoms, etc.), that span organisational boundaries, that satisfy the demands that different kinds of organisations may place upon it and that can evolve with the changes in organisations and their demands? In particular we can identify a number of areas which highlight some of the subsidiary issues that arise in this context?

Reaching agreement in large, multi-cultural, competitive groups is difficult. The larger the group, the more problems there will be before agreement can be reached.

A durable commitment to adhere to an agreement is desirable, however, circumstances change, as do people's minds. Procedures for accommodating such changes in developed systems in some ordered fashion need to be instituted in order to maintain or modify assurances concerning the purpose and non-operational requirements of the system.

Policing an agreement means that it is necessary to create, fund and trust an authority to carry out that policing. Failure to monitor, or if discovered deal with, those that break a commitment makes it impossible, in any assured fashion, to provide the required guarantees.

If there is no support in this area then competitive advantage will only be derived from the ability to deal with situations in which no or only a limited agreement is possible.

Conflicts also need to be identified. For instance, there may be a conflict between requirements for openness (open architecture) vs. confidentiality requirements (security) which requires identification and resolution.

4.7 Accommodating different interests

A design process or technical framework has been identified within which a real distributed system could be designed and implemented. The user-community could either be partitioned into sets of concerns, or, for example, by the nature of the tools used. It is already clear that such constituencies do not in general have a one-to-one correspondence with the set of projections and models. That is, although the model framework is a powerful analytical tool it does not partition the sets of players. For example, information modellers clearly model an information system for an enterprise, so that it would appear to be natural to talk about the duality of enterprise and information models. What information modellers do, in fact, is to build an information model of an enterprise with half an eye on enterprise, computational, engineering and technology issues. Of course, it may be a policy (enterprise) goal to abstract completely away from, say, the technology. However, according to our way of thinking, this would be an enterprise issue concerning the information model and not an enterprise model issue. The set of models remaining in each projection after abstracting away all other issues is the set of concepts which we could have arrived at by simply considering the cognitive framework.

5 Enterprise concepts and concerns

5.1 Introduction

This chapter introduces the set of concepts which are used in chapter 6 which shows the ways in which they inform the process of building an enterprise model. These concepts have been derived from the preliminary analysis of enterprise issues. There are a number of other concepts, notably those concerned with value, used in enterprise modelling methods which are only lightly touched upon here.

The concepts explored here may be used in two ways. The first is as background to the process of eliciting the nature of the problem and defining its boundary. Awareness of the concepts allows this process to reach sufficient depth.

Concepts specify the basic elements from which a design can be constructed. Rules state how the concepts may be combined: they constrain the freedom a designer has to make design choices. A design or system which violates an architectural rule or contains concepts that cannot be expressed in terms of the basic concepts, may operate as required, but will be more difficult to fit with other designs or systems that do not violate any rules. It is not mandatory to follow the recipes or guidelines provided by methods and tools, but their use is likely to reduce the design effort.

Although there are tools which aid the recording of information about the problem area the skills exhibited are largely obtained through experience and are not formalised. Once the problem has been defined, however, the process of modelling can make more formal use of the concepts.

5.2 Conversations

Our basic requirement is to provide a structure to describe and model purposeful interaction. The ANSA computational model describes an interaction model which lays down the rules governing the invocation scheme between two objects. One object, the client, wishes to make use of a service provided by another object, the server. It is assumed that the service provided by the server in this invocation is known to the client. In the enterprise architecture we can use these ideas but the context is extended to sequences of exchanges where information flows to some purpose. These sequences are labelled conversations.

In talking about conversations we are focusing on how interactions within our systems are concerned with the dynamic build-up of ‘status’ and relationships. That is with ‘changes in the world’ which stands behind trading, securing, and other activities.

In the context of ANSA conversations will be mediated by computer systems and thus the interfaces will need to be consistent with the nature of these conversations.

A conversation, therefore, is a sequence of exchanges which is intended to fulfil a specific social goal (e.g. validating and authorizing a new computer user). In building an enterprise model such exchanges are used to elicit the domain of interest, the entities and their relationships within the domain, non-operational requirements as well as goals. The conversation is the largest unit of communication in our model.

The exchanges are characterised by locutionary, illocutionary and perlocutionary acts which describe different senses of utterances. These descriptions derive from linguistic philosophy [SEARLE69] and from linguistics [BROWN83][LEVINSON83]. They are generally subsumed under the name of speech acts though sometimes “illocutionary acts” and “speech acts” are used as synonyms [see §5.3].

A speech act pattern is a sequence of speech acts that forms a logical unit in a conversation. The notion of speech act pattern is crucial, since it is used to validate that a conversation is coherent and complete. The relations between the various illocutions that together form a speech act pattern is defined by a conversation program. These programs not only include references to linguistic structures but also involve empirical knowledge such as the current states of the speakers, hearers and possible worlds. Conversation programs include procedural rules that define the flow of control and actions taken during each stage (see below) in a conversation. They define what speech acts may be performed under specific conditions.

Speech acts in a pattern can be grouped into sets of alternatives, depending on the actual conduct or execution of the pattern. There might, for example, be separate validation procedures depending on whether the candidate for validation is a British citizen, an EEC resident or a foreign national. These alternatives form a stage in a conversation. The set of alternative stages obviously depends on the type of the conversation.

Moves are speech acts that activate stages in a conversation. They thus control the flow of the conversation process through its various stages.

5.3 Speech acts and illocutionary logic

5.3.1 Speech acts

A speech act is the basic unit of communication. The three main categories are propositional or locutionary acts, illocutionary acts, and perlocutionary acts.

A propositional act simply expresses a fact, such as “It is Christmas Day today” or “There'll always be an England”. Propositional acts can be evaluated to be true or false (though it may not be easy to determine which, as in the second of our examples). Nothing more is said about how the evaluation is performed, nor about the theory of truth being assumed. Propositional acts can meaningfully be uttered by people or machines.

An illocutionary act, or illocution, is a basic unit of human communication; it is always performed when a person utters certain expressions with an intention - for example, “I promise to write the letter” or “I refuse to pay the bill”. When the intention has been recognized by the hearer(s), the illocutionary act has been successful, and we say its meaning has been understood. Questions of the truth of an illocution do not arise; rather, the act creates a commitment that in a moral sense binds the future behaviour of the

parties and pledges them to certain activities or expectations. Illocutionary acts can be expressed through mechanical means as well as vocal means. They are, however, always an expression of human concern or intention.

A perlocutionary act is an act that produces effects on the feelings, attitudes or behaviour of the hearer(s), for example to get someone to write a letter on request. Again, truth of a perlocutionary act is not an issue; success is, and occurs when the perlocutionary act has its desired effect. Perlocutionary acts are not further considered here.

5.3.2 The structure of an illocutionary act

In general an elementary illocutionary act has a propositional content and an illocutionary force which is some part of the utterance which indicates the nature of the fit between the propositional content and reality [SEARLE69]. These two components can be distinguished in any meaningful utterance. An illocution also takes place in an illocutionary context. All these constituents are important in understanding the meaning of an illocutionary act.

The content refers to the propositional act embedded in the illocution. For example, the content of “I refuse to pay the bill” is “I will not pay the bill”, considered as a (true or false) statement of future fact.

The force is the fundamental component of an illocution. It determines the social relationships or commitments established and the way in which the content is related to the world. Illocutionary force is generally implicit in the utterance. If the illocutionary force is made plain, for example in uttering a promise or a bet, the utterance is described as explicitly performative.

The context of an illocution is defined by the speaker, the hearer(s), the time and place of utterance, and the set of possible worlds. The first four are self-explanatory; the fifth is the set of the speaker's and hearers' contexts of interpretation.

The point or purpose of a speech act is called by Searle its illocutionary point ‘which is part of but not the same thing as illocutionary force’ [SEARLE79].

5.3.3 Types of illocutionary acts

There are a number of categorisations of illocutionary act. An example with just five categories, is as follows:

Category	**objective**	**examples:**
assertives	to say how the world is	to state, to predict
commissives	to commit the speaker	to promise, to agree
declaratives	to change the world by saying so	to christen, to deem
directives	to get the hearer to do things	to order, to request
expressives	to express the speaker's attitudes	to apologize, to condole

The analysis based on an example of categorisation has some system consequences in that it enables distinctions to be made between states of affairs and obligations.

Other categorizations are possible. Another, also due to Searle, replaces assertives by representatives which commit the speaker to the truth of the propositional content.

5.3.3.1 *Propositional content conditions*

The content of an illocution cannot be arbitrary, but must satisfy certain propositional content conditions. For example, the propositional content of the sentence “I order you to draw a triangular square” cannot be satisfied. These conditions are analytic conditions (i.e. ones which can be determined from consideration of the propositional content alone).

5.3.3.2 *Preparatory preconditions*

Preparatory preconditions specify the state of affairs that a speaker must presuppose to exist in the world if an intended illocutionary act is to be performed. For example, in placing a purchase order the buyer presupposes that the supplier has not gone bankrupt, has the same address as earlier, and still sells the ordered products. These conditions are synthetic, i.e. they depend on empirical facts.

5.3.3.3 *Rule-governed behaviour*

Distinctions are made between rules which order behaviour, say traffic regulations, and those that describe the way a particular action may be constituted. The latter are called by Searle constitutive rules. Together with the preparatory preconditions and the propositional content they may used as a schema to analyse intentions. An example of a schema and notation for an office based context is in [AURAMAKI88].

5.3.3.4 *Metaphors*

Certain utterances may have more than one meaning - a literal meaning and a metaphorical meaning. Using the terms outlined here these utterances will have primary and secondary illocutionary points. In some cases the propositional contents will be opposed and in others the point may be implied rather than stated. In stating intentions and the promise of achieving certain goals utterances may also contain propositions described elliptically or by means of hints. Schemas developed based on speech-act theory will need to be aware of these ambiguities.

5.4 The importance of conversations

The approach to examining the process of developing systems that we are adopting is to examine the structure of the exchanges or conversations in which the system specifications are elicited and the organizational requirements captured. An approach using speech act theory shifts the focus of attention away from formal reasoning as the basic cognitive activity to the space of cooperative actions undertaken by the enterprise as a whole.

This is essentially a social activity. It represents a shift in emphasis to the actions by which groups originate and coordinate a network of interconnected activities and obligations, and which considers the commitments that individuals as agents impose on each other and on the system. It does not deny the use of logic in analysis and decision making.

The following features of a speech act formalism are important:

1. The description of the **purpose** of the conversation

   (i) Experience of the use of speech acts indicates that in order to determine whether a particular conversation is well formed (with respect to its social function), it is necessary to make certain

distinctions, e.g. between requesting and responding to a request, between agreeing to an offer and making a counter offer, between committing a transaction and aborting a transaction. All of these distinctions are distinctions of purpose.

(ii) Purpose is described in terms of the so called “illocutionary points” of the individual speech acts. It is an application dependent decision as to what different sorts of illocutionary point there are.

2. The description of the **conditions** for successful conversation

(i) Any conversation can only be held on the basis of certain assumptions, which are described in the so called “preparatory conditions” for a speech act. Of course the complete set of assumptions can never be completely stated, since it is probably not finite. However, the more important assumptions, which in general will be those about the assumed impossibility of certain failure modes, should be made clear.

3. The emphasis on describing the **understandability**, **coherence** and **completeness** of the conversation

(i) Understandability, completeness and coherence of a conversation are expressed in terms of the structuring of a conversation into speech act patterns. Conditions and constraints such as sequencing, reference to context, the expression of alternatives, and control over dialogue structure form part of this structure.

(ii) The collection of the individual speech acts together with the description of the superimposed structure is called the conversation program.

(iii) Representations of the individual speech acts in the program can be installed as data items in some database. The structure can then be represented as a set of relations over those data items, and can therefore (as is desirable) be kept separate from them.

4. The ability to take a more complete view of a **policy** statement

(i) The reason for examining speech acts at all is that they, and they alone, establish the basis on which obligations (accountability, responsibility) are established. It is these obligations that are fundamental to a statement of policy.

5.5 Agency, agent, activity and resource

An activity is the set of substantive acts required to achieve some goal. The statement of the substantive acts and of the goal will be discovered through conversations, or other statements, or examination of existing systems and procedures. The activity will have a relationship with resources which may be created, destroyed or transformed. Agency indicates the function of an agent with respect to an activity and has a sense which implies responsibility as well as an active operation. These concepts are placed in a more formal relationship in chapter 6.

5.6 Constituencies

The notion of communities or constituencies is that of deciding the extent of the domain of interest and the set of entities within that extent. The ODP Enterprise Language includes a note that 'delineation of authority' has the potential for forming the basis of a rule. It is linked in the document with 'domains', that is with the transfer (of ownership, obligations), and with organizational structures. A problem related to deciding on an extent is the standpoint of the various **stakeholders** (see §5.6.1).

If we consider that activities take place in both the 'environment' and the 'system' then the distinction which defines the boundary is that activities in the environment simply occur and give rise to actions or events at the boundary. The system, on the other hand, is a set of activities where there is some control or policy of dealing with events from both outside and inside the boundary.

A point to consider is that to a large extent the behaviour of the 'environment' is known but cannot be changed although there can be radical effects at the system boundary viz attacking aircraft and defence missiles. These interactions may be completely novel i.e. not foreseen, or they may occur probabilistically from a set of possible worlds.

A boundary could therefore be defined as a predicate that distinguishes in a model those entities whose behaviour is directly observable and those whose behaviour can only be inferred.

This definition preserves the words system and environment which carry with them a number of preconceptions. It includes implicitly the notion of an observer who could be placed within the 'system' part the 'domain of discourse' (which is one way of viewing the world), or the observer could be in one system viewing interactions at the boundary with another system, or from a point of view outside the domain of discourse (or outside the two systems).

5.6.1 Stakeholders

The notion of an observer can be extended to the idea of a number of different relationships within the domain of interest. These stakeholders are assumed to share the same domain of interest. It will be part of the process of problem definition and the discovery of the boundary to ensure that they have this domain in common out of the many possible set of domains they inhabit.Thus communities can be characterised in the following way;

- System/problem owners,
- Solution providers,
- Application designer,
- Distributed infrastructure designer,
- Service provider,
- End-user.

Similar lists of stakeholders, with the same objective, are used in practice. An example is the ICL OPENframework which has Enterprise Management, Users, Application Developers and Service Providers as types of stakeholders.

The various stakeholders have differing concerns which must be identified and correlated with other concerns. They are an essential part of the process of problem definition and of the delineation of boundaries.

Techniques are needed to cope with multiple views about real-world entities and the multiple views that can be held of the information recorded about these entities and the processes they undergo. The idea of ownership and all that entails is an essential component of idea of stakeholding along with that of selection and choice.

The ANSA approach is to take a 'contractual' view of ownership. The legal view contains a large number of concepts and provides a suitable basis describing rights of ownership, its transfer and the resolution of disputes. Part 2 of the Reference Model of Open Distributed Processing [ISO92a] includes a definition of contract as part the set of ODP architectural concepts.

Selection of design options is offering designers a choice. Offering a choice injects a sense of self determination and hence of freedom and individualism which accords with the sociotechnical approach. Selection is a form of decision making. Others include more complex forms of negotiation and political machination.

5.7 Structures

The structure of an organization can be regarded as the set of all tangible and regularly occurring features which help to shape its members' behaviour [CHILD84] [DAWSON86]. It is a socially created pattern of rôles and relationships which exist within it. This encompasses both formal and informal structures. Organizational structure can be regarded as a tool serving two purposes: firstly, structure provides the basis for coordinating the diversity of tasks and activities which are performed by members of the organization; secondly, it helps control individuals or groups by guiding or forcing them to act in ways determined by others and which reflect dominant interests of one or more groups.

Different approaches to the theory of organizational structure exist. The following is an approach which could be useful in the design of information systems. It views the organizational structure from several perspectives. It is important to note that these structures show considerable overlap. The overlap may extend beyond the boundaries of the enterprise especially, for example, in the case of informal social structures or the membership of professional bodies.

5.7.1 Control structures

Control structures provide the means by which goals are established and monitored. This results in structures that reflect the strategic, managing and operational requirements of the enterprise, where:

- the division of labour is made according to the tasks, responsibilities, obligations and accountability of individuals,
- the number of hierarchy levels, the span of control in the hierarchy and the designation of formal reporting relationships is made according to perceptions of efficiency and suitability.

5.7.2 Information Structures

Information structures will also follow the strategic, managing and operational requirements of the enterprise as they provide essential

information about the functions employed to accomplish enterprise goals. They include:

- the design and incorporation of systems to ensure effective communication, integration and participation
- the design and incorporation of systems for performance monitoring, appraisal and reward.

5.7.3 Organizational structure

The organizational structure will depend not only on the control and information structures but on notions of specialisation and efficiency as, for example, where specialist support is deployed in a matrix fashion. The structure, in general, will be formed by:

- the grouping together of individuals into units and the grouping of these units into further units according to functional requirements.

5.7.4 Social structure (for work related issues)

ANSA recognizes the existence of informal structures and the implications which these may have on certain design issues. For example, in designing a secure system it would be quite misleading to assume that people only interact through the formal channels of communication. The confidentiality of information has as much to do with the people using a distributed systems as with the system itself. The informal channels will have to be dealt with outside the system by enunciating a policy with regard to breaches of security, e.g. the possibility of punishment.

Organizations are special kinds of social groups with a defined power structure, defined responsibilities and with accepted collective goals, for example, profitability. Although goals, responsibilities and the power structure are defined they can be illusory and systems accommodating the needs of organizations need to cater for both the formal and the informal structures.

5.8 Organizational structure and distributed systems

An analysis of organizational structure may reveal issues which may be pertinent to the distributed system designer. A description of classes of organizations in terms of power structures, for example, distinguishes between federated and hierarchically structured organizations. A hierarchical organization has a central point of authority whereas a federated organization will have more than one point of authority, none of which can force a decision on the others.[1] The hierarchical or federated model of authority will require different design decisions. The centralization or distribution of certain functions in the computer system may mean that it is not suited to certain models of authority such as hierarchical or federated structures.

The power structures of organizations may differ quite considerably and the architecture must accommodate such diversity. For example, in a banking environment, the authorizing and executing bodies should be kept separate to

1. Strictly speaking this structure is that of a confederation. Federation usually implies that some authority is centralised though possibly with safeguards to ensure consensus.

avoid misappropriation of funds. A similar situation arises in the separation of the legislative and executing branches of government. In a military application the opposite is often the case - the flexibility required on the battle ground often dictates that the authorizing and executing bodies be the same. There are situations in which the normal chain of command is broken and no centralized authority can be consulted, making it necessary to allow low level executing bodies to seize control. In such cases it is sometimes necessary to allow the same rôle holder to exercise two different rôles and sometimes it is necessary to prevent it. Similarly in a planning environment such as a CAD/CAM company, the authority to modify plans could rest with the designer, the implementor or both. Either way the authority should be allocated explicitly and mechanisms may be necessary to help in preventing the misuse of authorization.

Another facet of organizational structure is the span of control and the number of hierarchical levels in an organization. The span of control refers to the number of subordinates a manager has reporting formally to him or her.

Tall and flat organizational structures are distinguished by the number of hierarchical levels which exist in an organization. There are many arguments for and against both tall and flat organizations. From the point of view of the designer it is important to know what the specific or range of hierarchical levels in an organization is to ensure that the system can provide the appropriate facilities. The organization should be aware that the incorporation of distributed information systems may allow faster and more efficient ways of collecting and processing information, thus possibly allowing the increase of the span of control.

Changes to the span of control are possible, for example, by delegating responsibility down to lower levels, thereby relieving the working load on higher levels of management. These managers may now have the capacity to cope with an increase in the number of subordinates reporting to them - thus increasing their span of control.

Another aspect of organizational structure which is linked to the span of control is that of horizontal differentiation between groups, sections, departments and divisions. Given the range of activities to be undertaken in an organization, the problem is how should the activities be grouped together in a manner most suitable to the objectives of the organization? There are a number of models available and each should be assessed in terms of its appropriateness to the organization and its objectives. The most commonly used models are the 'product' and the 'functional' models. The functional principle groups members of an organization whose activities are aimed at producing different products but who use the same resources or share similar expertise in their activities.[1] The product principle groups members of an organization who may be functionally different but are contributing towards developing and making the same product or service.

In an organization, if people are grouped together under a functional principle, than it is possible to have a large number of them reporting to one manager thus encouraging a large span of control causing a flatter and smaller management structure. If, on the other hand, an organization consists of a large number of different and highly specialized departments, for example

1. This area is commonly referred to as 'business process engineering' especially when information technology plays a significant part in the restructuring [DAVENPORT90].

with distinctly different products, then the dissimilarity is likely to encourage a small span of control resulting in a taller structure.

Distributed system designers should be aware of these different models and the extent to which their systems can be configured to accommodate one or the other.

5.9 Processes

Processes can be characterised in terms of description, consumption and transformation. Description, together with the recording of actions, is the essential characteristic of information processing systems. Transformation is concerned not only with the change of state of information as maintained by information processing systems but must correspond with transformations, or changes of state, in the real world e.g. bank balances and changes made during the application of manufacturing processes. These correspondences of the records of an information processing system with the consumption of consumable and non-consumable real resources requires certain guarantees in order to be effective and secure. Abstract processes have already been touched upon when discussing the idea of stakeholding and stakeholders. They are concerned with such activities as negotiation, arbitration and making decisions.

5.9.1 Business process

Business process is concerned with the characteristic ways that organisations conduct their affairs. Examples may be taken from the manufacturing, service and government sectors. Business processes may be also be composed of lower level business activities which may be organised in libraries. While business processes may be characterised at a sector level and provide a market for specialist providers with sector knowledge and experience, business activities are basic and much more general building blocks out of which business processes may be constructed. The importance for enterprise modelling is that any agreement on the validity of such analysis provides opportunities to employ notions of re-use and will generally aid the development of interworking in federated systems.

Such analyses are related to ideas of metrics and the methods favoured by some management consultants. They include notions of mission statements, critical success factors and value chains. These ideas will also be used in that part of the modelling process which determines the domain of interest as well as providing some metrics aiding procedures for total quality.

5.9.2 Abstract process

Distributed systems, especially those which incorporate legacy systems, will have individual domains of authority. If joint working is to take place between such domains then some process of negotiation between the domains is required. The negotiation can take place in various ways, for example, externally to the two systems or by some process of delegation. Mechanisms are needed to allow the decisions to be incorporated into the system. In general the systems developed in ANSA for the management and monitoring of distributed systems have a suitable set of concepts and mechanisms to tackle this problem. These allow the negotiation and the subsequent incorporation of decisions to be made either as a delegated process or by

appeal to system owners. The basic mechanism is described in [APM.1018.1 93]. The problem for enterprise modelling is to be able to provide descriptions of the required behaviour which can be incorporated into implemented systems and which is modifiable in the light of any subsequent renegotiations.

5.10 Contracts

Contracts express statements of promises to act in certain ways and statements about the consumption of resources or the communication of information. They are also believed to express commitment. When a commitment refers to the actions of agents they express an expectation. In speech act terms they are commissives. As well as agreements, contracts also carry with them the idea of obligations on the agents who are parties to the contract. An example of the use of notions of contract is in the telecommunications industry where they are used to label management interfaces to systems. These interfaces will include both informational and functional aspects.

5.10.1 Obligation

Obligations, as well as permission and prohibition, extend the functional and informational aspects of agreements to include ideas of authority and responsibility. Obligations can be described using a modal logic which allows the basic concepts to be manipulated. The manipulations that can be described using such a logic include the constitution of an obligation, its modification and fulfilment. The uses of such logic not only include the specification of desired behaviours but their post-hoc verification [SANTOS91].

5.10.2 Autonomy and authority

Notions of authority are closely related to the granting and rescinding of permission. Authority can, therefore, be described and manipulated using the logic noted above. Authority is also used in the discovery of boundaries when deciding on a domain of interest during the problem definition phase of system development. Autonomy describes the situation where there is no delegated authority, or at least no acknowledged delegated authority.

5.10.3 Responsibility, accountability and liability

Responsibility is the state of delegated authority. It is also used in the same way as authority in discovering the problem area boundaries. Thus, as with authority, it is related to structural and organisational issues. Authorities in this sense are not necessarily disjoint and may therefore require contractural obligations reflecting the integrity of the overlapping domains of authority.

Accountability and liability are particular attributes which in contractual terms describe how to deal with a breach of obligations. Together with the main outcome they form part of a contract. There is a strong sense in which they are an analogue of one of the uses of terminations of the ANSA computational model.

5.11 Policies

Policies are part of an administrative process where decisions points are generally unclear and where the exit from the process is also unclear.

A statement of a policy, resulting from this process, can be defined as consisting of:

- the **objective** - the definition of a future state of affairs desired and believed to be feasible. It includes a set of objects to which the policy is directed or applies as well as the set of objects who initiate or carry out the missions.
- **missions** - a set of statements which define a set of actions intended to achieve the objective
- **constraints** - a set of predicates over behaviour which indicate what shall and what shall not be done in undertaking some implementation of the mission statements.

Policies need to be distinguished from **plans** and from the statement of missions which are a part of a policy. Plans are derived from policies by a suitable management process (see §6.3) and are prescriptive whereas missions express intentions. The management process will clearly identify the decision points and exit points so that the general management activity can be carried out.

There are a number of points that can be made about the way policies and policy making is viewed. For the most part the usages blur the distinction between policy and plan. That is the distinction between administrative process resulting in a policy and the management process by which plans are derived and the operational process of carrying out the plans.

A common usage is one where policy is attached as a label for a particular field of activity, which also may be coupled with an expression of general purpose or of a desired state of affairs. Statements of policies which make specific proposals or which prescribe particular outputs are closer to the notion of plan than policy. Policy can also be used to describe authorisation, although the statements are generally contingent and unclear.

5.12 Designing the organization

The main questions which relate to the concept of organizational design are: how is the organization likely to change as result of the introduction of new technology, and how can the introduction of new technology be used to create opportunities for positive change.

For example, the introduction of computer technology is seen by many as an opportunity to deal with complexity and uncertainty because such systems can facilitate the collection, interpretation and analysis of hitherto disparate information. The question that arises is to identify the organizational and management structures that are best suited to exploit such a tool.

In general, the structure of an organization is seen as a set of complex entities and relations which can be modified to take many different forms. The introduction of information systems, and in particular of distributed systems, may be regarded as a useful tool in helping organizations restructure themselves.

Many different questions can be asked when redesigning the organization; the following set of questions relate to the aspects of organizational structure cited in §5.8:

- with regard to functional and structural rôles: how should the work be broken down - for example, whether specialization should be preferred to integration. Specialization has its benefits but also incurs higher cost in terms of communication. Integration, on the other hand, may provide different workers more scope and responsibility. Another issue is how much discretion should be given to people and how much control should be exercised
- with regard to the type of management hierarchy: options concerning whether the organization should be tall or flat in terms of levels of management and spans of control. Such decisions have implications for communication, motivation and implementation costs
- with regard to the grouping together of individuals into units: the grouping can be done by functionality, product or by geographical location
- with regard to the level of coordination and integration: should different parts of an organization be forced to integrate or should more autonomy be given to different departments?
- with regard to the way in which management should be structured to maintain adequate control: should the new technology be used to centralize or decentralize decision making?

Most design decisions cannot be made purely on technical grounds; they require human judgement. Sometimes a design cannot include particular judgements but must accommodate a whole range of judgements by incorporating and automating policies. While the designers may be charged with the inclusion of the policies in the design, they may not be in a position to make decisions about the actual policies to be employed and will need to consult others. Examples of the elements of the system for which policies may be needed include the power structure, the value system, the security system, performance assessment, reliability measures, the policing system, penalty imposition, arbitration over ownership, safety systems, training schemes and consideration of the ethics of individuals.

Because organizational policies can be interpreted as special requirements which are determined by people outside of the design team, it is important that the designers spend time to understand fully all of the policies. The policies themselves may clash, and to provide a system-wide resolution each specialized policy must be defended. These factors point to the need for special responsibilities. Specific responsibility can be allocated, for example, for designing a safe system, for designing a reliable system and designing a secure system.

6 Building an enterprise model

The previous chapter has considered a number of concepts concerned with the approach to problem definition. The ideas of interaction, boundaries and the process of policy definition are all connected to understanding the limits and concerns of the system under discussion. Ideas of organisation were introduced which emphasised notions of context when dealing with sets of concerns, especially when there is overlap between them.

This chapter is concerned with further concepts about structure and construction of enterprise models which lead ultimately to the implementation of systems. It can be considered as a complement to the reference model published by the Object Analysis and Design Special Interest Group of OMG [OMG-OAD92]. The latter was developed to provide a comparison of a number of commercial analysis and design methods. It divides the development process into a number of well-understood modelling phases and provides a set of concepts arranged in a hierarchy to aid this comparison. The modelling phases are:

- strategic
- analysis
- design
- implementation.

All of these phases contribute to the full definition of the system and all are informed by general notions of object modelling which enables a consistent view to be taken of the phases. These four phases are supplemented by further phases of construction and delivery. The strategy phase determines the purpose and extent of the system, commonly as a set of requirements together with clustering concepts which may lead to a family of models.

The set of concepts developed in this chapter considers an abstract enterprise model where three types of basic entities are considered and the relationships between them analysed.

This chapter also uses the structure used for the entities of the abstract enterprise model to explore the notion of a policy driven system.

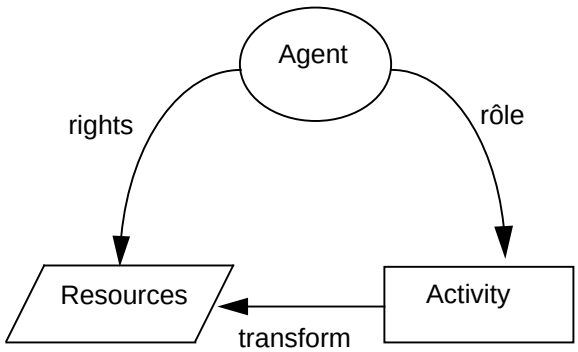
6.1 Enterprise model entities

The basic component of the enterprise model takes the form of an entity-relation schema defining three sorts of entities: **activities, agents,** and **resources:**

- *Agent*: This is a name attached to a set of responsibilities, e.g. customer fault manager. An agent must always be an appropriate answer to a Who? question

- *Activity*: An activity is to be distinguished from the performer of the activity. Thus an activity is an answer to a What? question, and takes the form of a verb, e.g. CustomerComplaintsHandling
- *Resource*: Resources are the answers to With? or ByMeansOfWhat? questions.Between these three types of basic entity there are six kinds of relation:
- The first three are illustrated in figure 6.1 and also indicate inter-entity relations:.

Figure 6.1: Enterprise model entities and relationships



Agent-Activity: The set of Activities with which an agent has some relation constitute the *functional rôle* of that agent;

- Agent-Resource: The relation between an agent and a resource is an access right, such as the right to create, to destroy, to allocate;
- Activity-Resource: Activity and resources are related by notions of creation, destruction, use and transformation. The basic relation in information technology terms is an that of an access mode, such as read or write (for information resources) or provide, consume or transform when describing Activities on commodity resources.

There are also intra-entity relationships as follows:

- Activity-Activity: Activities interact with each other. Such interactions are usually mediated by the exchange of resources, though direct interactions can also occur.
- Resource-Resource: The relation between resources is what in information processing terms is called the **conceptual schema**.
- Agent-Agent: The set of agents with which an agent has some relation constitute the *structural rôle* of that agent

Many existing approaches to the design of systems do not treat these basic concepts in a formal or precise manner. Being able to do so improves the quality of systems as they are better able to absorb the culture of the contexts in which they will be used.

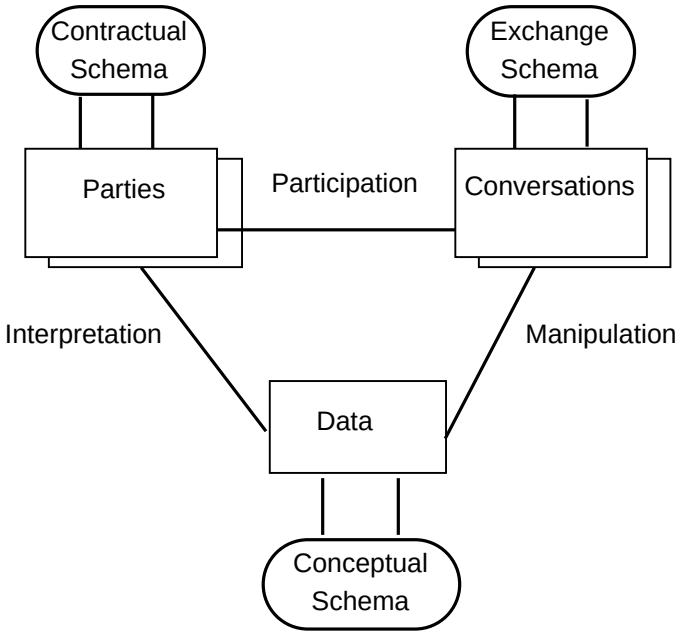
6.2 Modelling conversations

The use of this model can be illustrated with regard to another set of entities which are more concerned with modelling the interactions

between people. (Though it just as well be applied to systems.) It uses the same entity relationship (E-R) notation to illustrate the approach and is strongly related to the construction of a database or information model. The particular context is therefore that of information modelling where the commodity that is being created, exchanged and transformed is information.

This model is shown in figure 6.2 and is illustrated and described as in §6.1.

Figure 6.2: Information language components and relationships



There are three main types of entity in the language we use to describe information: *Data, Conversations* and *Parties.*

1. *Data*: These are conventional file- or database structures. (ANSA does not prescribe any particular file or database technology.)
2. *Conversations*: These provide the context in which interpretation of data takes place
3. *Parties*: These are the entities that take part in conversations; they correspond to the *agents* in §6.1.

Between these three types of entity there are six possible relations:

1. *Data - Data:* This is what is often referred to as the *conceptual schema*, which defines the constraints on the relationships between data items. The notion of the conceptual schema, in the database world, represents a view of the universe of discourse which remains largely invariant [ANSI/SPARC75]. It therefore requires, in the terms used here, an appropriate definition of the boundary of the domain of interest and the set of entities within that boundary and their relationships. The other parts of this schema model are a internal schema which maps the conceptual schema to the underlying support mechanisms and access methods and an external schema which allows a variety of application views using suitable tools. The

main motivation for this division was a principle known as data independence which allowed applications to be re-developed and new ones added without need to change the underlying model. In a distributed environment these classical database ideas need to be extended and this is explored in chapter 11.

2. *Party - Party: T*he relation between parties is described in terms of the rôle relations between them, e.g. customer-supplier, client-server, colleague-colleague etc. The set of these relations is called the *contractual schema*: it constrains the structure of the relationships between parties.
3. *Conversation - Conversation:* Different kinds of conversation exist: they are characterised as types of speech act, such as, for example, *command*s and *requests*. Conversations can refer to each other, and one conversation can be dependent on another. The set of relations and dependencies between conversations is called the *exchange schema*; it defines the constraints on the type and structure of conversations.
4. *Party - Data*: Parties both access and interpret data. Rules can be expressed stating which parties are allowed to access and interpret which data (as opposed to merely accessing it). For example, a secretary who reads a letter on behalf of a principal must be given access to the data, but the secretary remains part of the access path; any interpretation of the data by the secretary has no validity or authority as seen by the organisation.
5. *Party - Conversation*: Parties participate in conversation. Rules can be expressed over which parties can participate in which conversations, and with what powers and authorities.
6. *Conversation - Data*: Conversations manipulate data and data structures. (Manipulation includes the passive case of merely referring to data.) This means that information systems also participate in conversations when a secretary updates a database for instance. This can be seen either as an abstract conversation between the secretary and a principal who later refers to the database, or as an abstract conversation between the secretary and the DBMS which would include such things as checks on the authority to access and update. Both views are, of course, equally valid; the relation between them is a relation between conversations and hence part of the exchange schema.

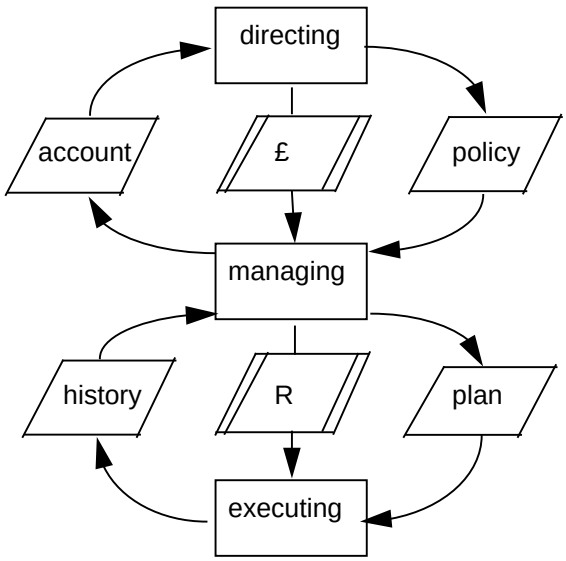
6.3 Policy driven systems

In any large system there will a large number of functional requirements. Methods have been developed to deal with this complexity and tools are available to support these methods. The idea of policy driven systems is to add a further analytical tool to the armoury of those being analysed by the OMG Special Interest Group noted above. That analysis is dealing with object oriented methods which can be applied to current non-distributed systems and to classical implementation systems, such as relational databases (see [RUMBAUGH91]). With distributed systems there will a greater degree of autonomy of developed systems which will be expected to cooperate with other autonomous systems. They will be guided in this cooperation by certain rules and policies. Such policies need to be incorporated into the systems and the functioning and adherence to them managed.

The basic abstract tool proposed to understand the importance of policy driven systems uses the concepts developed in previous sections and is based on concepts of management and monitoring developed in ANSA [HOFFNER92].

The starting point of the management of a distributed system is seen as support for the management of the enterprise. Manageability in this sense is a property imparted by an architecture[1]. What are the properties of an architecture, or at least of its enterprise and information projections, required to achieve this manageability?

Figure 6.3: Policies, plans and execution



The theory of management embodied in the enterprise projection is expressed in terms of the direction/management/execution trichotomy (figure 6.3). This structure is related to a similar partitioning of an enterprise into strategic, tactical and operational used as an expository device by management scientists. Figure 6.3 shows three main activities or functions that of directing, managing and executing. There are a number of conversations taking place concerning policies, plans and resources.

1. Directing->Managing: policies
2. Directing->Managing: allocation of generic resources
3. Managing->Executing: plans
4. Managing->Executing: allocation of specific resources
5. Executing->Managing: records of executions of plans and resources used
6. Managing->Directing: accounting for the resources allocated

In the context of the particular type of enterprise modelling considered in this document, the most important policy about which conversations take place is the IT strategy policy. This policy is reflected in the other conversations.

1. This account draws heavily on conversations with Mike Martin of MARI and John Dobson of the University of Newcastle

The manner in which policies and plans are expressed is different. In the case of policies it will be necessary to show that a policy is consistent and in the case of plans that they can be executed. There is a further requirement which is to show that plans are consistent with policy.

Policies may be divided into:

1. procurement policies;
2. operational policies, and:
3. decommissioning policies.

This division creates sensible obligation boundaries and interfaces. It can also be used to provide internal structure of the system management agency (e.g. by considering the different types of generic resources involved).

In particular, it is useful to distinguish between (a) resources as the subject of plans and policies; and (b) resources as the means of realising plans. The first is concerned with operations management and the second is concerned with procurement and the investment process. For IT systems, change management is a combination of procurement and decommissioning, and therefore decommissioning policies need also be treated along with procurement in the investment process. Procurement involves issues such as the proper use of investment and risk/benefit management, together with operations management in the context of distributed systems.

Issues of proper use and of conformance to specification may be considered to be a technology issue though the statements made about them use enterprise architecture concepts.

6.4 Taxonomies

The idea of communities or constituencies was introduced in chapter 3. The taxonomies in this section are used to illustrate how the set of people that use an information system may be structured into organisational, social or other types of groups. The identification of different groups is important specially when a (distributed) computer system is to be used by members of different groups as a means of inter-group communication. For example, a system may be set up to relay order, price and delivery information about certain products between two companies. The system spans organisational boundaries and this will require special design consideration to identify, document and safeguard the contractual relationship between the organisations.

Large distributed computer systems are increasingly crossing organisational, political, economic, and social boundaries. The identification of these boundaries helps in the identification of agents and their contractual relationships. This in turn helps the application designer make informed decisions about the way in which applications should be structured.

In the very simple example above for instance, it would not do to place all information about orders in one place, accessible to all clients, so they can work out for themselves what delivery times might be. Instead the delivery time calculation should be performed by the suppliers and only the results should be made available. However, if the relationship amongst the clients and supplier were inter-departmental within the same company, then clients can be allowed to see one another's orders. This information may even be used to negotiate order priorities between clients departments for instance.

There are many kinds of taxonomies that can have a bearing on the design and operation of an information system. The common notion of community may be used to characterise a body of people that have something in common, be it organisational, financial, economical, political, municipal, social, or physical. Although is may be possible to discern many or all of these communities and taxonomies in a single system instance, they are not necessarily the same (congruent).

6.5 Particular design concerns

When considering the various communities involved in solving and developing the solution to a large problem their various concerns need to taken into account and understood so that the various models developed have suitable points of correspondence to each other. The achievement of such correspondence may turn out to be difficult to accomplish in any formal fashion, especially because of difference in attitudes and in the languages used. Some of the concerns exhibited by the different stakeholders are noted.

6.5.1 System/problem owner's concerns

There may be multiple views of a system and possibly of the components of the system. The administrator regards a test tube as a resource that costs money, the consultant thinks about the specimen carried by the test tube and derives his diagnostic information from it. The abstract models we have used to introduce the basic enterprise concepts have used the entity-relationship notation. This provides a static data-driven approach though in practice the elicitation process usually employs an implicit process model. The process issues are left to the application designer. With distributed systems there is a need to complement the data-driven approach with stronger process notions. The move to object-oriented design methods requires the support of object-oriented development systems and databases.

A small example from hospital administration illustrates the multiple views that can be held on a patient record. The hospital administrator requires a factual record, whereas the consultant treats the record as an aide memoire as well (a sort of personal notebook). The fact that both needs are catered for by the same patient record is a matter of convenience and perhaps historical. rôle and agent ideas (as well as identification of need) are important in deciding how to structure such a record.

6.5.2 Application designer concerns

The application designer has a special requirement to validate his design with other stakeholders and therefore he has a special responsibility to explore the relationship to other stakeholders. This means, for example, that the satisfaction of end-users is of concern to the application designer.

The application designer needs, therefore, to have representations of the design which allow communication with other designers, system owners and end users. This implies the use of methods and tools e.g. OO design and CASE tools [OMG-OAD92] which allow the different representations to be manipulated as well as the progression of the design from informal to formal specification (see chapter 3).

The enterprise architecture should allow a designer to choose alternative solutions and allow consideration of:

- the control and management of resources;
- questions about flow controls and scheduling of services.

The enterprise architecture concept of negotiation provides the basis of allowing choice in design decisions. Giving people choice injects a sense of self determination and hence of freedom and individualism [BAUDRILLARD88].

6.5.3 Enterprise language

In the RM-ODP the prescriptive model [ISO92b] uses a set of languages, including an enterprise language, to specify ODP functions and to describe their rôles in ODP systems. These languages use the modelling concepts of Part 2 of the Basic Reference Model of Open Distributed Processing [ISO92a]. These modelling concepts are based on object modelling notions. The reference model categorizes modelling concepts as follows:

1. **Basic interpretation concepts:** concepts for the interpretation of the modelling constructs of any ODP modelling language.
2. **Basic linguistic concepts:** concepts related to languages; the grammar of any language for the ODP Prescriptive Model must be described in terms of these concepts.
3. **Basic modelling concepts:** concepts for building the ODP Prescriptive Model; the modelling constructs of any language must be based on these concepts.
4. **Specification concepts:** concepts related to the requirements of the chosen specification languages used in ODP. These concepts are not intrinsic to distribution and distributed systems, but they are requirements to be considered in these specification languages.
5. **Architectural concepts:** structuring concepts that emerge from considering different issues in distribution and distributed systems. They may or may not be directly supported by specification languages adequate for dealing with the problem area. Specification of objects and functions that directly support these concepts must be made possible by the use of the chosen specification languages.

The RM-ODP enterprise language is used in the specification of an ODP system. The specification defines the requirements and policy of the ODP system.

7 Summary - enterprise projection

This document has presented a consolidation of the purpose of the enterprise architecture by suggesting the definitions of a number of enterprise concepts. These concepts are considered to be a sufficient starting point to provide a context in which enterprise models can be validated. It should be pointed out that the context in which the enterprise architecture has been described is that of the development of information processing systems and that this is not the only context to which the concepts can be applied. The place of tools and methods within the enterprise architecture has also been partially explored.

7.1 Purpose of the enterprise model

It is now widely recognised that one of the major problems associated with the design of large distributed systems that serve organizational needs, is that the structure of the IP system cannot be separated from the structure of the organisation it serves. There is thus a need for a set of concepts which enable the organizational environment of a system to be thought about as carefully and rigorously as the IP system itself. These concepts must be sufficiently generic to be capable of satisfying two sets of needs:

1. to model an organisation and the changes in it that the IP system must assist (or will induce)
2. to model a system in such a way that it is possible to reason about the extent to which it will in fact assist or induce those changes.

The issues in the enterprise architecture are thus essentially structural: does this arrangement of computer hardware and application programs match the structure and purpose of the organisation it is intended to serve? This is the sense that is used to describe the conformance of the enterprise model.

This document identifies several concepts used to model an enterprise. Whilst the process of enterprise modelling and of the development of associated tools and methods itself falls outside the scope of ANSA, it is important for information processing systems designers to understand these concepts so that informed decisions can be made about the way in which systems and applications should be structured, thus ensuring that the systems they build will suit their purpose.

The verification and validation of the concepts has not yet been completed. The relationships between the concepts and their relationships with concepts in other projections, as well as the relationship with existing and desirable tools has not been fully investigated either. A number of the concepts have found their way into the ODP working drafts where they have received some measure of

independent validation and acceptance. This process is continuing as the concepts are refined through application.

7.2 Enterprise concepts

Three main sets of ideas have been introduced:

1. conversations
2. structures
3. taxonomies

The enterprise architecture provides a language and a structure for considering and reasoning about purposeful interaction. Conversations have been used to provide a model of interaction and this concept is crucial to the whole architecture. Structures have been used to introduce the ideas of boundaries to define systems and to point out how in distributed systems the problems to be considered in crossing these boundaries. Taxonomies have been introduced to in order to identify the rôles of the various stakeholders of a particular system.

The sets of basic concepts and taxonomies are expected to be particularly useful in aiding the elicitation and problem definition process.The concepts will also help designers make decisions about the way in which a system or application should be structured (distributed) such that the resulting system or application fits the environment in which it will be used.

8 The information projection and model

8.1 Objective

The following chapters discuss the concepts needed to do successful information modelling for ODP systems.

The objective is to enable people to assess information modelling techniques and tools in the context of ODP, so that they can choose an appropriate tool or technique from the many that are available.

This document points out only the major issues. Further work would be required in order to provide more detail or to recommend specific choices.

8.2 Overview

The information projection and information model have been identified as part of the ANSA Framework (see chapter 2). The purpose of the *information projection* of a particular system is the identification and location of information and the description of information processing activities.

The *information model* is a set of concepts, rules, recipes and guidelines, which allows the designer to

1. model the use of information as the representation of obligations that exist between agents in an enterprise and as an instrument which mediates those obligations,
2. model the use of information as interpretation of data,
3. describe the components of a particular system in information terms.

Just as Enterprise is concerned with purpose and purposeful activity, Information is concerned with **meaning** and **meaningful activity**. In ANSA, the primary focus for describing and reasoning about activity is at the places where objects interact. Therefore the ANSA Information Model is particularly concerned with **meaningful interaction**.

8.3 Relationship to ODP standardisation

ANSA Information Model ideas have been transferred into the joint ISO/IEC/ CCITT work on Open Distributed Processing which is currently in progress. The concepts and terminology used in this document are aligned with the usage in ISO 10746-3 *RM-ODP: Prescriptive Model*[ISO92b] and in particular with clause 6 which defines the Information Language.

Where terminology has been adopted from ODP, this is pointed out in the text. Not all of the ideas presented here have yet been transferred to ODP and so some additional terminology has been introduced.

Work is still in progress in ISO/IEC and CCITT, in particular on the definitions in ISO 10746-2 *RM-ODP: Descriptive Model*[ISO92a] and the languages in ISO 10746-3 *RM-ODP: Prescriptive Model*[ISO92b]. When this is complete it may be possible to adopt further ODP terminology.

9 Information Model requirements

Scarrott observes that there is no generally agreed answer to the simple questions about information: “What is information?”, “Has information natural properties?” [SCARROTT88]. Scarrott concludes that the most basic function of information is to control action in an organised system. This conclusion, together with Scarrott’s definitions of ‘organised system’ and ‘information’ will be taken as a starting point.

An **organised system** is an interdependent assembly of elements and/or organised systems.

Information is that which is exchanged between the components of an organised system to effect their interdependence.

Systems that are not ‘organised’ are outside the scope of this discussion, and therefore ‘system’ will mean ‘organised system’ in the following discussion.

9.1 Representing obligations

Where an obligation exists between agents in an enterprise, the agents are interdependent elements, and the obligation is their interdependence. It is the exchange of information between the agents that mediates the obligation.

Obligations in an enterprise exist to control the actions of the agents. This is consistent with the idea that the purpose of information is to control action.

9.2 Interpretation of data

In order to mechanise the exchange of information, the information must be turned into some representation which can be transmitted mechanically (sound waves, electric signal, digital bit pattern). Such a representation is **data**.

At the receiving end, the data must be turned back into ‘information’. The relationship between data and information is thus maintained by sender and receiver independently.

It is possible to describe a system in terms of the relationships between information which is intended to be conveyed, the data which is actually conveyed, and how that data is interpreted. Such a description helps to avoid the problems that can arise due to both technological and organisational differences between sender and receiver.

Technological and organisational differences arise in distributed systems because there are many kinds of equipment, and equipment is owned and operated by different organisations. The differences cannot be eliminated. There is no authority with the power to enforce the required global rules, even if such rules could be formulated.

The issues of representation and interpretation are considered in more detail in chapter 11.

9.3 Describing of components

The information model is concerned with how the components of a system can be described, and how meaning is assigned to those descriptions. The issue is the relationship between a system and a model[1] of that system.

The information issues in the construction and interpretation of models of systems are discussed in chapter 10.

1. Model is used here in the sense of a “simplified description of a system to assist calculation or prediction” [COD82].

10 Representation and interpretation

This chapter introduces concepts that can be used to find and analyse information issues. The issues are:

- how systems are described,
- how those descriptions (models) are interpreted, and
- how common features can be identified.

These issues are closely related to the purpose of the system and the purpose for which a model is required which are enterprise issues.

10.1 Systems and models

A model is constructed to be a representation of a system and therefore, when interpreted, gives information about the system. The most basic information issue for a model is that there must be a way to interpret it. The model is interpreted in order to extract information about the system it represents.

The system that is modelled need not exist. The model may be intended to give information about the benefits and the problems that might arise if the system it represents were to be constructed.

ANSA is concerned with systems that are themselves used to model other systems. Many computerised information systems exist in order to contain a model. For example, the purpose of a stock control system is to contain a model of how many of each item are in stock. The system being modelled might be a supermarket where you could count how many tins of beans are on the shelves. The model would contain a number that represents how many tins of beans are in the shop[1]. A manager might use the model to decide when to order more beans. The manager has interpreted the model to derive information that has determined action in the system being modelled.

A system that is used to model another system may itself be represented by a model. The interpretation of the model may be applied recursively. Interpretation of one model gives information about another model, which may be interpreted in its turn.

Some systems contain models of themselves. A simple example occurs in many hotels. Posted on the walls at various places is a plan of at least part of the building showing the fire exits. The plans exist to provide information that guides the actions of the guests in the event of a fire.

When a system contains a model of itself, the model may be of the system as it was at some time in the past. It may be a model of how the system ought to be,

1. In practice, a stock control system would model how the system has evolved over time. This would be used to forecast likely future behaviour and action would be taken on that basis. The basic idea remains the same. The model is interpreted to give information that controls action in the system.

or is believed to be. In these cases, the system being modelled and the system containing the model are different, even though they are closely related. Altering a building does not in itself cause the fire escape plans to change. Explicit action is required to bring the plans up to date.

ISO 10746-2 *RM-ODP: Descriptive Model*[ISO92a] defines the concepts '**template**' and '**instantiation (of a template)**' so that a template is a model of some objects. For an object that is an instantiation of the template, the template is a model of the object as it was when it was instantiated.

A system can contain a model of itself as it really is. The actions carried out by a computer system are controlled by a model which is represented within the system. Since the actions of the machine are controlled by the programs and data, it is interpreting[1] these as information. Since the system is controlled by the model, changing the model changes the system.

For many computer systems, the underlying model and the system's ability to change itself are well controlled. Many programming languages make a sharp distinction between code, which cannot be changed, and data which can be changed. Some machines have hardware mechanisms that can enforce this distinction. Causing data to be interpreted as code is an explicit step that makes subsequent changes to that data at least difficult if not impossible. In ODP, this explicit step would be instantiation for templates that specify that part of the state of an object is fixed.

There are some environments which do not make such a sharp distinction between code and data. Smalltalk 80 provides some facilities of this kind, other object oriented programming languages such as CLOS provide even more. The ODP definitions accommodate such systems since an object is characterised by its state, and its state determines its behaviour. There is no requirement that any of the state of an object be fixed, the template from which the object is instantiated determines whether or not there is any fixed state.

When a system is used for modelling, the system's ability to change itself must be taken into account when deciding how to interpret the model.

10.2 Structure and compositionality

The interpretation of a model can be made easier by constructing the model as a collection of parts and relationships between those parts.

An object-based model is composed of **objects** that represent pieces of the system, and **relationships** between those objects. When interpreting the model, relationships between objects correspond to inference rules. The meaning of a collection of related objects is inferred from the meaning of the objects by applying the rules corresponding to the relationships.

Being able to infer the meaning of something from the meanings of its parts according to rules specified by the relationships between the parts is known as **compositionality**.

1. The machine is 'interpreting' the data in the sense that its actions are controlled by the data. The machine does not assign meaning to the data but is following a path specified by a programmer who assigned meaning to the data that was anticipated.

The final outcome of the interpretation should not depend upon the order in which the inferences are made. Being able to draw conflicting conclusions implies that the inference rules are unsound.

The ability to compare alternatives depends upon being able to relate models so that the parts that are the same can be distinguished from the parts that are different. Part of one model being the same as part of another model means that they represent the same part of the system being modelled.

The notion of 'sameness' is quite difficult. Some of the problems are identified in the schema integration work of Batini et. al. [BATINI86] and the multi-database work of Litwin et.al. [LITWIN90]. Names map to entities in contexts; the meaning of a name is captured in the mapping. This topic is discussed further in §10.3.2.

10.3 Representation, data and naming

Information is an abstract concept that must be represented when it is exchanged between entities. Assigning meaning to a model means that the elements of the model represent information. In both these cases, there is a symbol that stands for the concept.

10.3.1 Relationship to naming

Applying the ANSA Naming Model [APM.1003.1 92], the symbol denotes the concept by standing as a name for it. The name must be bound to the concept in some context, the information is captured in that binding.

What are thought of as symbols in one model may themselves be concepts that are represented by other symbols. For example, 57 may stand for the number of tins of beans on the shelf. The symbols '5' and '7' presented together in that order are commonly accepted to represent the number fifty seven. In a machine, the number may be represented by the pattern 00111001, which is itself represented by electrical signals. That pattern in a machine also commonly represents the character '9', the 'meaning' depends on the context in which that pattern is used.

Our ability to represent and manipulate information depends upon many agreements that have evolved over a long period of time. Where there is no existing satisfactory agreement, parties wishing to communicate must form a new one, usually built on top of some existing agreements.

The ANSA Naming and Information models provide the basis for understanding how to form an agreement on the meanings of symbols that can be exchanged, and therefore, how an interaction with symbols can be meaningful.

10.3.2 Identity and sameness

A naming domain determines a set of entities that are distinct. Each entity has a distinct identity (e.g. it can be pointed at) but this does not necessarily mean that there is a test for sameness or for distinctness.

It is difficult for a conversation to take place between entities which are using different naming domains. This is true of both human and mechanised conversations. Someone who does not distinguish between different kinds of

fruit will not be able to understand a conversation about the relative merits of apples and pears.

For a model, the naming domain defines which parts of the system can be distinguished in the model. A stock control system that does not distinguish between brands cannot be used to determine which brand is the most popular.

Deciding what is to be considered the same and what is to be considered different is a fundamental part of developing an information model. Developing a naming model that can accommodate all the distinctions that are likely to be required is an important part of the information design for a system.

10.3.3 Name translation

Comparing names is straightforward but comparing named entities is often difficult or impossible.

In a model used for prediction, the entities are in the future. In many other cases the entities are concepts or beliefs that are intangible and so not amenable to any direct tests. Even if there is a way to test the entities themselves, the purpose of the model may be to avoid having to manipulate the entities directly.

For all of these reasons, the issues involved in comparing names rather than named entities must be understood.

A name denotes an entity in a context. Name comparison has meaning only when the names are interpreted in the same context. Comparing entities usually starts with two names which may be in different contexts. This leads to the question: “What is a name in this context for the entity denoted by that name in that context?” Note that there is a recursion issue here in that it is necessary to identify the contexts.

It may be necessary to find some other context in which to perform the comparison. Reaching that context from the original contexts may require translations through yet further contexts.

If there is a context in which both entities have unambiguous names, and names for the entities in that context can be found, then a meaningful comparison of the names is possible. The names must be unambiguous if having the same name is to mean being the same entity. If the context is known to contain no aliases then having different names means being different entities.

10.3.4 Coexistence of views of sameness

In distributed systems, the indirection inherent in potential remoteness can be exploited to give a variety of useful mechanisms. For example, in-service upgrades can be provided by replacing one object with another without the old clients being aware of the change. A client may use a service without being aware that it is provided by a group with an evolving membership. Resources may be conserved by passivating and activating objects without their clients being aware of the changes.

A common theme in all these examples is that there is one point of view from which it appears that something has changed, and another point of view from which there appears to be no change.

Not only is it necessary to have different notions of sameness in models of different systems, it is also necessary for several notions of sameness to coexist in a single model.

The designer must specify what is meant by “same” for each usage of the concept.

10.4 Classification

Having introduced the idea of similarity, the next step is to identify collections of objects that are alike. Classification is a well established way of managing the complexity of information. A collection of entities can be considered as a class once some characteristic that they all share has been identified. It is also usual to define the class as all entities that have that characteristic[1].

Since some characteristics imply other characteristics, a classification scheme based on those characteristics will include a hierarchical relationship between classes. Choosing the characteristics according to some principle gives rise to a taxonomy.

There are many different taxonomies that may be used to classify the same collection of objects. Each taxonomy is appropriate to some purpose. Since there are many possible purposes for classification, there will be many different criteria for determining the appropriateness of a taxonomy. There can be no taxonomy that is best for all purposes.

Different organisations have different needs and so may choose different taxonomies for classifying things that are of interest to them. If the organisations wish to interact meaningfully, it will be necessary to reconcile these differences. This is an example of federation. The participants must negotiate an agreement that allows them to interact meaningfully.

Two common taxonomic principles are classification by ancestry, and classification by observable characteristic. Although there is often a correlation between the taxonomies, it is important to remember that they arise from different principles and an inference drawn from one classification scheme cannot necessarily be applied to another.

For example, the debate about inheritance and subtyping in object oriented programming illustrates the problems that can arise. Classification based on inheritance gives one taxonomy, classification by supported methods and their parameter types gives another. These classifications are useful for different purposes, and asserting that either one or the other is the right and only way to classify objects will make some issues much harder to address.

10.5 Requirements for modelling

A modelling technique suitable for ODP must satisfy a number of requirements.

1. The definitions of **type** and **class** in the *Basic Reference Model of Open Distributed Processing - Part 2: Descriptive Model* attempt a more precise definition of the concept of class. Unfortunately, the definitions do not specify that the condition for membership of a class must be based on a characteristic of the entity itself. The ODP definition of class includes but is not limited to classification in the sense required here.

- It must be able to represent both pieces of systems and relationships between those pieces. Since both are important, it should not put undue emphasis on one or the other.
- It must be compositional in order to support analysis, refinement and the relating of models with similar structure and elements.
- It must allow the notion of “sameness” to be defined, and also allow different views of sameness to coexist and be related.
- It must allow multiple classification schemes to be used.

The common theme of all these requirements is that the modelling technique must not prejudice the designer’s ability to deal with heterogeneity.

11 Meaningful Interaction

This section considers how information is exchanged between elements of a system.

11.1 Interaction concepts

The system is modelled as a collection of objects related by their interactions. The emphasis is on mechanised objects that interact according to the ANSA computational model.

Note that although 'object' is still used to mean a distinct element in a model, in this section it will be an element that manipulates information rather than an element that embodies information by standing for a concept.

Interaction is by exchanging symbols. Meaningful interaction requires that the interacting parties have compatible interpretations of these symbols. In terms of naming, the interacting parties exchange names so the bindings of the names must be equivalent. The contexts in which the names are generated and resolved have an intersection which includes the names that are exchanged.

The naming model can be used to explain the processing of information. Information is passed from one object to another when the sender names something in a context shared between the objects.

Name translation is another important element of information exchange. An object can translate names if it can find a name for an entity in one context given a name for that entity in another context. Such an object can pass information between objects that cannot do so directly. Such translations are very common. The interpretation of the information model in terms of naming provides a basis for reasoning about the correctness of the translation.

The exchange of symbols can be mechanised and modelled in several ways: shared data, shared objects or interaction.

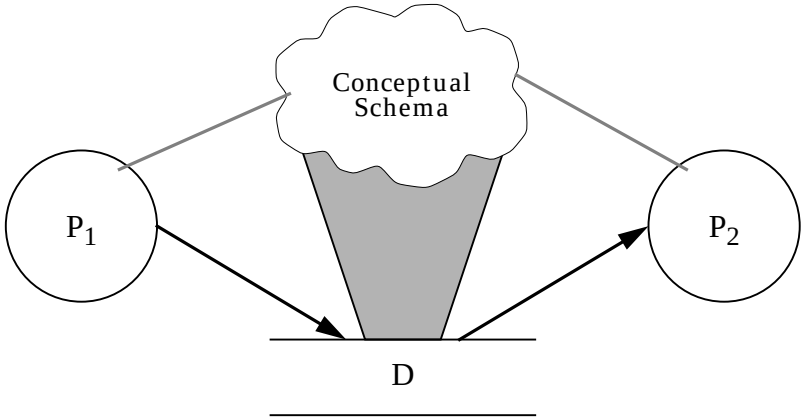
11.2 Shared data model

Conventional 'information systems' consider data and the conceptual schema (traditional database design) alone (figure 11.1).

Party P_1 can communicate with party P_2 by making a change to the data D. P_2 can subsequently read the data and observe the change. The conceptual schema provides a common interpretation of the data and thus helps to ensure that the communication is meaningful.

In this view of systems, data is shared and can be accessed by several parties. By changing the data (updates) the parties are able to communicate.

Figure 11.1: Conventional database model



To ensure that changes to the data are interpreted by the reader in the same way as intended by the writer, (modifier), reader and writer agree to common interpretation by means of a shared conceptual schema.

Since the data reflects 'reality' (e.g. the number of tins of beans on the shelf), the conceptual schema 'defines' the link between the data (structure and relations) and reality.

In general, it is assumed that there will be many parties that access the data and all of these parties must agree to interpret the data in the same way. The conceptual schema is shown as being associated with the data since it is a common view to which all accessors must subscribe.

11.3 Object based model

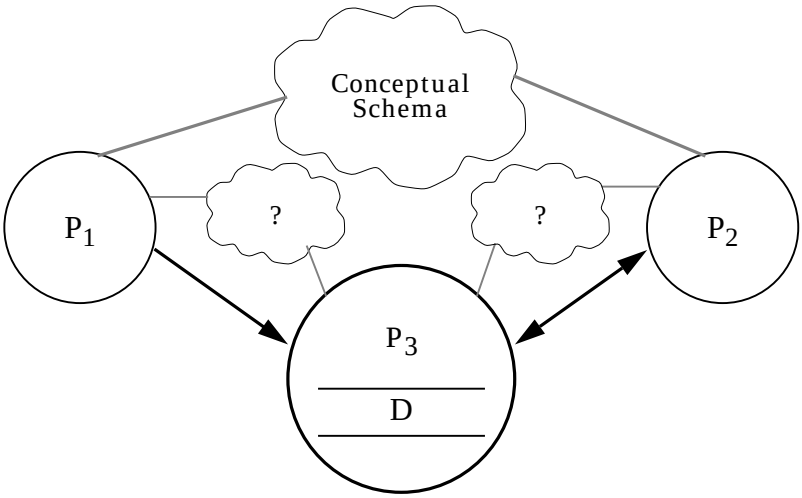
Shared data is often implemented as shared memory. The data is considered to be a passive resource manipulated directly by the parties. In a distributed system, that approach breaks down. The desire for data independence also leads to the hiding of internal structure behind an interface.

In an object based model, such as that adopted by ANSA, the shared data is replaced by a shared object that encapsulates the data. The shared object is equivalent to another party. Interactions between a party and shared data in other systems become interactions between parties in ANSA (figure 11.2).

If P_1 and P_2 are to communicate using P_3 as an intermediary, they must agree upon a view of the universe of discourse as represented by the conceptual schema. P_3 need not adopt that conceptual schema in the same way as do P_1 and P_2. If P_3 simply accepts symbols and passes them on, it need not subscribe to any particular view of what those symbols represent.

The conceptual schema is associated with the shared object that encapsulates the data in the same way that the conceptual schema was associated with the data in the shared data view. All those parties (objects) that use the shared object as an intermediary must subscribe to the same conceptual schema. Just as the shared data did not interpret itself, so the shared object need not interpret the data it encapsulates.

Figure 11.2: Object based model



Even if the shared object is simply forwarding symbols, there is a relationship between the shared object and the other objects that interact with it. There are information issues associated with that relationship. Preserving the meaning of the indirect interaction illustrated above depends upon the relationships between the shared object and the objects that interact with it.

P_3 is an example of an object that participates in more than one conversation. In RM-ODP terminology, objects have a **behaviour** which is defined by a **dynamic schema**. It is the behaviour of the object that relates the conversations in which it participates.

11.4 Conversations

The example in §11.3 illustrated how an object based model can be used to mimic the shared data model. If the object based model is taken as the starting point, then that example is one of many possibilities to be considered.

The basic form of interaction in an object-based model is between two parties. In this case, it is necessary to consider:

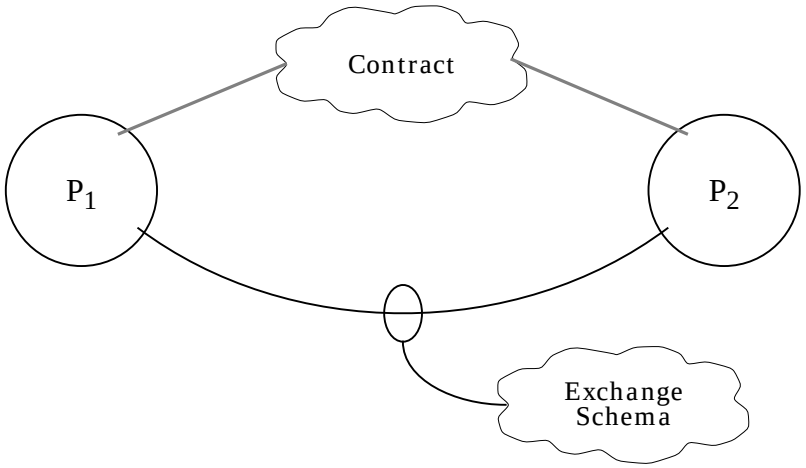
- the contract between the parties (which is described in enterprise terms)
- the **conversations** in which the parties participate.

Conversations have a structure which is shown as described by an Exchange Schema in figure 11.3.

In ANSA, the structure of conversations between mechanised parties is governed by the computational model, in particular by the computational interaction model. A conversation consists of invocation requests and responses. Each request consists of an operation name and zero or more parameters. Each response consists of a termination name and zero or more parameters.

In the ANSA computational model, the parameters are references to interfaces at which objects provide services [APM.1001.1 91]. The information model view of a computational interface is a point at which an object is willing to participate

Figure 11.3: Relationship between parties



in a conversation. The exchange schema of the information model corresponds to the interface type system of the computational model.

In the computational model, each interface has a type which is defined using elements provided by the interface type system. Similarly, the information model exchange schema provides elements used to define conversation types.

Parties that interact will engage in one of a range of possible conversations that are composed of an agreed set of elements with agreed meanings. The set of elements and their meanings is defined by the conversation type.

Although conversation has been explained as the way in which meaning is assigned to computational interactions, the concepts apply to any kind of interaction. In general, a conversation is how objects exchange information in order to affect each other's behaviour. Therefore, the required relationship between the behaviours is sufficient to specify the conversation.

11.5 Interaction and conversation

11.5.1 Direct and indirect interpretation

The computational model defines the parameter passing model as giving the ability to pass the use of an interface. In terms of the naming model, parameters are invocation names for interfaces. A parameter intended to convey information must be interpreted. Adopting the pure computational model view, parameters are interpreted indirectly by invoking operations.

Operation and termination names are interpreted directly. The operation name in a request determines the action in the server which receives the request. The termination name in a response determines the action in the client that receives the response. This means that it is the operation and termination names that are used to exchange information directly.

Meanings must have been assigned to all the operation and termination names. The names are usually chosen so that they can be considered as words in a natural language. Names chosen in this way have a connotation. A person, such as a programmer, who sees the name will think of the word and

its common usage. This will be taken to be at least a partial definition of the meaning of the name in the conversation type.

A commonly used simple example is of a stack with operations named “push” and “pop”. From a computational point of view, these names are no better or worse than any others. It can be argued that from an engineering point of view, the names “a” and “b” are better because they can be exchanged more efficiently using a simple representation as character strings. In this case the information issue of having a useful connotation is usually given preference over the other considerations.

11.5.2 Interpreting responses

If a client is to interpret a response, it must be able to interpret some part of that response directly. It can invoke the result parameters but this still leaves the problem of interpreting the response to those invocations. The termination names in the ANSA computational model provide a way to end that recursion since the names are symbols that can be interpreted directly.

An alternative approach is to have another kind[1] of parameter which can be interpreted directly to control the subsequent action of the client. Any such parameter must be a symbol that has been given a meaning in the conversation type. There are no difficult information or engineering issues in adopting this approach. The assignment of meaning, and the transmission of an appropriate representation, can be done in the same way as for operation names. The difference is that, for such parameters, there is no predefined way to link the distinct symbols to distinct actions.

It can be convenient to use a symbol to transmit something that has a definition in terms of its response to invocations. In such cases, the direct interpretation of the symbol must match the indirect interpretation through interactions. This is straightforward for a simple interpretation of constants such as strings and integers.

Some models introduce primitive types, such as Boolean, whose values have at least been given a connotation, although it is not necessarily linked to the information issues of the particular system in question. The types can be considered primitive if the only way to distinguish them is by some mechanism that could not be constructed from whatever other mechanisms are available. For example, in many languages there is a conditional construct which is the fundamental way to distinguish boolean values, and this may be the only way to choose alternative courses of action.

11.5.3 Relationships for direct interaction

The participants in a conversation will have some purpose for conversing, and this will be described in enterprise terms. There will be a contract that defines the obligations of the participants. Part of the process of establishing that contract is to agree upon the meanings of the elements of the conversation, or at least to establish a way to arrive at that agreement.

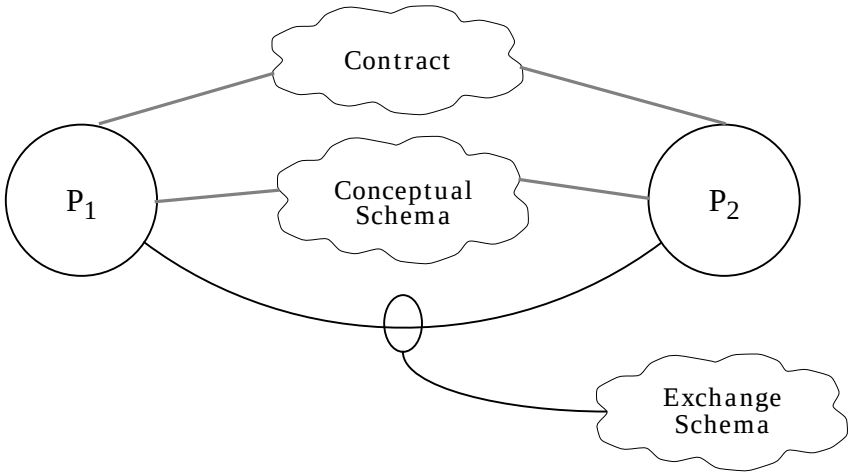
The participants must agree on a model for at least that part of the system about which they intend to converse. That common model is the conceptual

1. Note that ‘kind’ here is a meta-type notion. Non-interface parameters would need their own computational type system, different from the interface types of the ANSA computational model.

schema, and includes the definition of the naming domain for the names that will be used in the conversation. In addition to agreeing on the model, they must agree upon how the conversation will be structured. The structure of the conversation is defined by the exchange schema.

The various agreements that need to be in place are illustrated in figure 11.4.

Figure 11.4: Meaningful interaction between parties



The exchange schema defines the names that may be used, the conceptual schema defines which names are used and to what they refer. Before the conversation can take place, it will be necessary to agree upon specific representations for the names, and the mechanism by which the exchange will take place. This requires a mapping between the information view and the engineering and technology views of the system, and will usually be by reference to existing technology and established standards.

For homogeneous technology and engineering, there is usually no difficulty in establishing this mapping. For heterogeneous technology or engineering, it may be necessary to use translators in order to preserve the information.

11.5.4 Relationships for indirect interaction

When objects use an intermediary in their interaction, there are information relationships that are not directly related to a conversation.

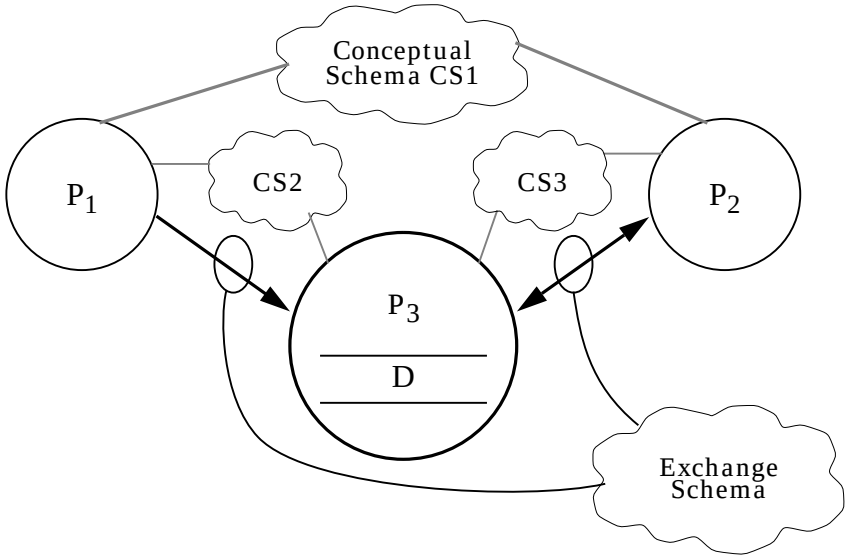
Figure 11.5 extends the idea illustrated in figure 11.2 in §11.3. It shows the conceptual schemata associated with the conversations which are governed by the exchange schema. It also shows the conceptual schema for the indirect interaction. (The contractual relationships are not shown, but there is a contract associated with each conceptual schema.)

The conceptual schemata CS2 and CS3 govern how P_3 interprets its conversations with P_1 and P_2. P_3 need not interpret the data in the same way as P_1 and P_2.

If P_3 is acting as a database, then the contracts will specify that it report what it has been told, rather than inventing answers at random. The conceptual schemata CS2 and CS3 will involve the concepts of ‘query’ and ‘update’ and will specify how those ideas are exchanged between P_3 and P_1 or P_2.

The idea that the response to a query is based upon earlier updates must be reflected in the behaviour of P_3. This would be specified in the dynamic

Figure 11.5: Indirect interaction relationships



schema for P_3. The appropriateness of P_3 for use as an intermediary is determined by relating its dynamic schema to the conceptual schemata.

Some of the data will be only partially interpreted by P_3. For example, CS2 may specify that some symbol in the conversation represents a number, but not specify what the number represents.

P_1 and P_2 also subscribe to conceptual schema CS1, and this is where any additional interpretation is specified. In particular, CS1 may specify that the pair <“Beans”, 57> is to mean that there are 57 tins of beans on the shelf, while CS2 and CS3 specify that a character string and a number are to be remembered and faithfully reported as a pair.

11.5.5 Conformance

Computational model type conformance requires that the operation and termination names be known to their recipients. It does not guarantee that there is any agreement upon what the names mean. The information model provides the basis for determining how such an agreement can be reached so that the interaction can be meaningful.

The information model requirement for type matching is that the participants in a conversation agree on meanings for the operation and termination names, and the meaning of the parameters. What it means for the parties to agree is closely linked to enterprise issues. Agreement is reached when the holders of the obligations are satisfied that agreement has been reached.

This means that there are no generally applicable rules for type matching in the information model.

Some conversations can be described entirely in terms of symbol manipulations without reference to the meaning of the symbols. This is often the case in conversations with intermediaries, as in the example in §11.5.4. This leads to the idea of defining the behaviour of objects, and the

conversations in which they participate, in terms of sequences of symbols that are exchanged.

Although it is possible to describe the behaviour of an object, or to describe a conversation in this way, it is hard to find any worthwhile classification scheme for such descriptions.

A particular problem is that describing the behaviour of an object is not the same as describing the conversations in which it may participate. An object may participate in many conversations. If different conversations are distinguished in the description of the behaviour then that description will be complex. This is especially true if there is no limit to the number of conversations in which the object may eventually engage. The object must also remember every conversation which is not yet known to be complete.

This suggests the other approach; conversations are required to be short. The extreme case is that each exchange is considered to be a conversation. This approach has its own disadvantage. Anything which cannot be done in a single exchange must be done as a collection of exchanges where one party is aware that they are related and the other is not.

The idea of a conversation made up of a number of exchanges but with a definite beginning and end is similar to some aspects of an atomic activity [APM.1004.1 92]. In the case of conversations between a server and several clients that also interact amongst themselves, the server's view of what has happened is different from any one client's view. The view of an observer following the activity is yet another different view.

Even for those conversation types that are amenable to a behavioural description, there is no known classification scheme that can support a conformance relation that would allow requirements to be matched against offers. The problem of the differing views means that it is not sufficient to be able to determine that behaviours are equivalent.

11.5.6 Classification by assertion

Since meaning, including behaviour, is important, it is necessary to find some way to match requirements against offers.

This can be done by identifying the concepts that are important and giving them names. Those who offer services can then name the properties that they claim for their services. Those who wish to use services can name the properties that they require.

For some cases, it may be possible to devise structured names and a relation that can be used as the conformance test. If this is done, it is important to remember that all that is being tested is an assertion about the service and an assertion about the requirement. The test also assumes that the relation does capture the appropriate notion of conformance.

The trading service provided by ANSAware includes mechanisms for asserting properties of services, and for finding services based upon asserted requirements [RM.102.00 92]. In that system, types are named and relationships between types are asserted by naming the types.

There is some checking of programs that trade services but many issues depend upon agreement between the programmers. There must be agreements between programmers who do not want their programs to interact as well as those who do. This is necessary to avoid ambiguities in the type naming context.

Services exported to the trader may also have properties associated with the offer.

The properties are presented as (name, value) pairs. The second component of each pair is described as a value because the trader adopts a partial interpretation of the symbol in the way described in §11.5.4.

For example, certain symbols are interpreted as numbers and the trader will perform numerical comparisons on that basis when dealing with import requests. The trader acts as an intermediary and it is up to the importer and exporter (or rather those people responsible for them) to have agreed upon the meanings of both the property names and the values associated with them.

12 Summary - information projection

12.1 Concepts

The Information Model is concerned with **meaning** and **meaningful activity**, and especially with **meaningful interaction** between objects.

Information is that which is exchanged between the components of an organised system to effect their interdependence, and its basic function is to control action.

Information is captured in bindings of names to concepts, both in representation as **data** for transmission, and in representing a system by a **model**.

Names do not have intrinsic meanings. Holders of obligations give meanings to names by making agreements. The agreements govern **conversations** in which the names are exchanged.

There is no general conformance relation for conversation types, but they may be named so that information about them can be exchanged. Those prepared to accept the consequent obligations may act upon asserted relationships between named conversation types, or the asserted type of a conversation.

12.2 Modelling requirements

A modelling technique suitable for ODP must:

- be able to represent both pieces of systems and relationships
- be compositional
- allow the notion of “sameness” to be defined
- allow multiple classification schemes to be used.

13 Relationship with other projections

The enterprise concepts should help designers in building up a picture of the structure of the enterprise in which the computer system is to be introduced or developed. The process by which such an understanding may be built up is outside the scope of ANSA. An enterprise model can be seen as providing a context in which a system is designed.

The enterprise architecture and the information about the relevant entities and their relationships should provide all the necessary input for a designer to decide how to distribute an application, or how to build an application from existing distributed components, such that the resulting system is fit for its purpose.

With this in mind, the set of things that can be discussed with respect to distribution are such things as:

- Who has the right to perform separation?
- Do the partitions follow organisational boundaries?
- Who has the right to define such boundaries?
- What goes where?

Analysis of these design issues should help assess whether the access, location and replication policies reflect the importance of the information and the roles of those who use it. For example, an obvious system design principle to place information close to those who need it most may be overturned by consideration of a security policy. It must also be possible to express the separation of parties from data, of conversations from one another, and of parties from conversations.

This then leads to decisions about the structure of applications in terms of object boundaries in the computational model. The computational model provides all the mechanisms to express distribution.

The engineering model offers a way in which the constraints due to a limitation of computing, storage and communications resources can be accommodated and the way in which separation and distribution are finally realised.

The technology model defines the process of conformance and whether the implementation meets the specification of the system.

References

[AIM92]

ANSAware 4.0 Implementation Manuals, Architecture Project Management Ltd., Cambridge (UK), (May 1992)

An Overview of ANSAware 4.0

ANSAware 4.0 System Manager's guide

ANSAware 4.0 Application Programming in ANSAware

ANSAware 4.0 System Programmer's Manual

[APM.1001.1 91]

"The ANSA Computational Model", APM.1001.1, Architecture Projects Management Ltd., Cambridge (UK) (August 1991)

[APM.1002.1 91]

"A Model for Interface Groups", Architecture Project Management Ltd., Cambridge (UK), (August 1991)

[APM.1003.1 92]

"The ANSA Naming Model", APM.1003.1, Architecture Projects Management Ltd., Cambridge (UK) (November 1992)

[APM.1004.1 92]

"ANSA Atomic Activity Model and Infrastructure" AR.004.00, Architecture Projects Management Ltd., Cambridge (UK) (February 1992)

[APM.1018.1 93]

"Management in Object-Based Federated Distributed Systems", APM/TR.039, Architecture Projects Management Ltd., Cambridge (UK) (1993)

[APM.1020.1 93]

"Abstract and Automate", APM.1020.1, Architecture Projects Management Ltd., Cambridge (UK) (1993)

[ARM 001.00 89]

"The ANSA Reference Manual", APM/ARM.001.00, Architecture Projects Management Ltd., Cambridge (UK) (1989)

[ARM89]

ANSA Reference Manual, Volume A, Part IV., Architecture Projects Management Ltd., Cambridge (UK), (July 1989)

[ISO92a]

Recommendation X.902: Basic Reference Model of Open Distributed Processing Part 2: Descriptive Model, ISO 10746-2, (November 1992)

[ISO92b]

Recommendation X.903: Basic Reference Model of Open Distributed Processing Part 3: Prescriptive Model, ISO 10746-3, (November 1992)

[OMG91]

“The Common Object Request Broker: Architecture and Specification”, Revision 1, Object Management Group,(1991)

[OMG-OAD92]

“Reference Model for object analysis and design. Draft 7.0", Object Management Group, (1992)

[OSF91]

“Introduction to OSF DCE”, Revision 1.0, Open Software Foundation, (December 1991)

[TR.017.00 91]

“An Application Programmer's Introduction to the Architecture, APM/ TR.017.00, Architecture Projects Management Ltd., Cambridge (UK) (November 1991)

[RM.102.00 92]

“ANSAware 4.0 Application Programmer's Manual”, RM.102.00, Architecture Projects Management Ltd., Cambridge (UK) (March 1992)

[ANSI/SPARC75]

ANSI/X3/SPARC,”Study Group on Database Management Systems: Interim Report 75-02-08”, *ACM-SIGMOD Newsletter,* 7(2), (1975)

[AURAMAKI88]

Auramaki, E., E. Lehtinen, and L. Lyytinen, “A Speech-Act-Based Modeling Approach”, *ACM Transactions on Office Information Systems*, 126-152, 8(3), (April 1988)

[BATINI86]

Batini, C., M. Lenzerini, and S.B. Navathe, “A Comparative Analysis of Methodologies for Database Schema Integration”, *ACM Computing Surveys* 18(4), 323-364 (December 1986)

[BAUDRILLARD88]

Baudrillard, J., “The System of Objects”, in *Design After Modernism*, ed John Thackara, Thames and Hudson, (1988)

[BROWN83]

Brown, G., and G. Yule.,” Discourse Analysis”, Cambridge University Press, (1983)

[CHILD84]

Child, J., “Organizations: a Guide of Problems and Practice”, Paul Chapman Publishing Ltd., (1984)

[CHECKLAND81]

Checkland, P.B., “Systems Thinking, Systems Practice”, Wiley, Chichester, England, (1981)

[COD82]

"The Concise Oxford Dictionary", Seventh Edition, Oxford University Press (1982)

[COMANDOS]

"A guide to the Comandos Platform". Comandos Consortium, (March 1991)

[DAVENPORT90]

Davenport, T.H., and J.E. Short, "The New Industrial Engineering: Information Technology and Business Process Redesign", *Sloan Management Review*, 11-27, Summer, (1990)

[DAWSON86]

Dawson, S., "Analysing Organisations", Macmillan Education, (1986)

[DRETSKE81]

Dretske, F.,"Knowledge and the flow of information", MIT Press/Bradford Books, (1981)

[HOFFNER92]

Hoffner, Y., "The Management of Monitoring in Object-Based Distributed Systems", to appear in *Proceedings of the Third IFIP/IEEE International Symposium on Integrated Network Management*, San Francisco, (April 1993)

[HUMPHREYS84]

Humphreys, P.C., "Levels of Representation of Decision Problems", *Journal of Applied Systems Analysis*, 3-22, 11, (1984)

[LEVINSON83]

Levinson, S.C., "Pragmatics", Cambridge University Press, (1983)

[LITWIN90]

Litwin, W., Mark, L., Roussopoulos, N., "Interoperability of Multiple Autonomous Databases", *ACM Computing Surveys* 2(3), 267-293 (September 1990)

[LYYTINEN87]

Lyytinen, K., "Different Perspectives on Information Systems: Problems and Solutions", *ACM Computing Surveys*, 5-46, 19(1), (March 1987)

[MONK88]

Monk, J., Y. Hoffner, "Accounting for human issues in the design of an architecture", in *UK IT 88 Conference Publication*, Information Engineering Directorate, Department of Trade and Industry, (1988)

[PIAGET78]

Piaget, J., "The Development of Thought", Blackwell, Oxford, (1978)

[RUMBAUGH91]

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorenson, "Object-Oriented Modeling and Design", Prentice-Hall International, Englewood Cliffs, New Jersey, (1991)

[SANTOS91]

Santos, F., and J. Carmo, "A Deontic Logic Representation of Contractual Obligation", in *First International Workshop on Deontic Logic in Computer Science* eds J-J CH. Mayer and R.J. Wieringa, Amsterdam, (December 1991)

[SCARROTT88]

Scarrott, G. G., “The Nature of Information”, The Computer Journal, 32(3), 262-266 (March 1989)

[SEARLE69]

Searle, J.R.,”Speech Acts”, Cambridge University Press, (1969)

[SEARLE79]

Searle, J.R., “Expression and Meaning”, Cambridge University Press, (1979)